8-2010

Service Contract Automation

Rico Knapper
Research Center for Information Technology (FZI), knapper@fzi.de

Benjamin Blau
Karlsruhe Institute of Technology, benjamin.blau@kit.edu

Sebastian Speiser
Karlsruhe Service Research Institute, sebastian.speiser@kit.edu

Tobias Conte
Research Center for Information Technology (FZI), conte@fzi.de

Christof Weinhardt
Karlsruhe Institute of Technology, christof.weinhardt@kit.edu

Service Contract Automation

Rico Knapper
Research Center for Information Technology (FZI)
Haid-und-Neu-Strasse 10-14
Karlsruhe, Germany
knapper@fzi.de

Benjamin Blau
Karlsruhe Institute of Technology
Englerstr. 14
Karlsruhe, Germany
benjamin.blau@kit.edu

Sebastian Speiser
Karlsruhe Service Research Institute
Englerstr. 11
Karlsruhe, Germany
sebastian.speiser@kit.edu

Tobias Conte
Research Center for Information Technology (FZI)
Haid-und-Neu-Strasse 10-14
Karlsruhe, Germany
conte@fzi.de

Christof Weinhardt
Karlsruhe Institute of Technology
Englerstr. 14
Karlsruhe, Germany
christof.weinhardt@kit.edu

ABSTRACT

Today's transition from a product- to a service-oriented economy implies fundamental technical, organizational and economic challenges. The trend of compensating missing core competencies by requesting business services from external providers to be integrated in internal end-to-end processes has recently gained tremendous momentum. Nevertheless, service level agreements between the parties involved are still specified for each service entity that is part of composite business services which results in a managerial overhead generated from multiple contractual relations. The contribution of this paper is threefold: *(i)* We analyze the fundamental requirements in the context of describing services, quality and agreements as well as their aggregation in a generic manner. Based on the results, we *(ii)* provide a holistic framework that enables the automation of service contracts for composite business services. Facilitating semantic technologies we provide means for describing service quality from a technical and business-oriented perspective, adequate metrics as well as quality aggregation operations in the context of composite business services. Furthermore, we *(iii)* evaluate our framework based on an industrial application scenario.

Keywords

Service Level Agreement, Indicator, Metric, Quality of Service, Service Composition

INTRODUCTION

Imagine a big sized company in the energy domain. The company provides a hotline for existing and prospective customers. Hence they need to deal with a business process *customer call* which can be simplified to five elementary steps: Retrieve call, register personal customer data, register customer issue, delegate issue to responsible colleague, end call. In order to handle the

process the company relies on several subsidiaries $p \in P$ providing IT services for internal usage.
The fundamental paradigm shift from a product- to a service-oriented economy implies novel technical and organizational opportunities. Service-oriented architectures (SOAs) enable the seamless integration of distributed services into end-to-end business processes. Companies tend to concentrate on their core competencies while requesting modularized business services from different service providers – and so does the big sized company in the energy domain. That is the business process host underlays various business process steps provided by functionality of appropriate services and diverse sources (cp. Figure 1).

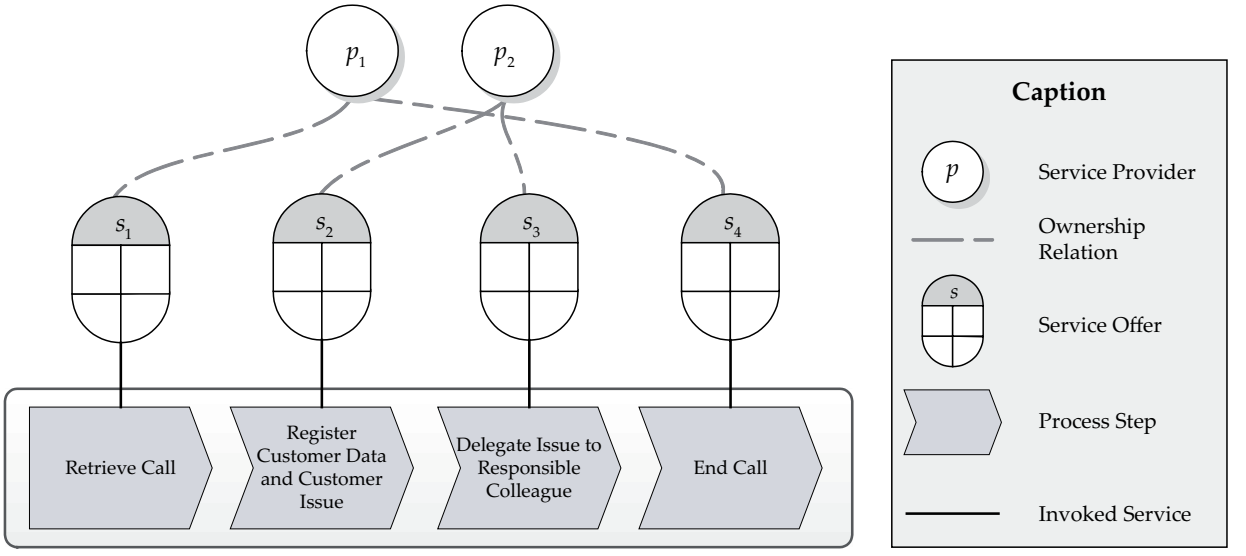

Figure 1: Business Process of the Customer with Invoked Services and Ownership Relations.

In the example, *retrieve call* and *end call* are provided by subsidiary p_1, *register customer data and customer issue* as well as *delegate issue to responsible colleague* are delivered by p_2. Our service consumer expects a service to function reliably and to deliver a consistent outcome at a variety of levels, i.e. Quality of Service (QoS). To ensure QoS, an agreement between the service provider and service consumer about the quality to be delivered must be founded on a legal basis, i.e. by specifying a Service Level Agreement (SLA). An SLA is a contract that defines mutual understandings and expectations of a service between service provider and service consumer (Jin, Machiraju and Sahai, 2002). Current state of the art implies that our service customer needs to negotiate Service Level Objectives (SLOs) based upon Service Level Indicators (SLIs) for every single invoked service. That is that the service consumer and the service providers need to identify the SLOs and SLIs concerning single services. For example, the second process step *register personal customer data* includes a customer relationship management (CRM) system for which the maximum downtime is one of the SLIs associated to the SLO "5 minutes at most per day". However, for the service consumer, it would be much more convenient to negotiate the SLA (i.e. SLIs and SLOs) on business process level. Thereby, the focus is put on the outcome or result of the process itself which actually represents the company's core competency. Moreover, a process-oriented service level agreement covering the whole business process including all invoked services reduces contractual relations as well as managing overhead along with that.

In order to implement the automated creation of such Process-Oriented Service Level Agreements (PROSA), a comprehensive methodology enabling a "Single Point of Contract" (SPC) is required (Knapper, Conte, Blau and Richter, 2010). To provide a comprehensive PROSA, aggregating SLIs and SLOs on a process level is a key challenge (Blake and Cummings, 2007; Jaeger, Rojec-Goldmann and Muhl, 2004; Unger, Leymann, Mauchart and Scheibler, 2008). Assume that concerning two invoked services (services s_2 and s_3) the single SLAs include identical SLIs (*maximum downtime* and *reaction time*). Aggregating *maximum downtime* on a process level requires a different aggregation algorithm than *reaction time* because of their different codomains. While *maximum downtime* is declared relating to time periods ("5 minutes at most per day"), *reaction time* is declared relating to metrical time specifications (e.g. "5 ms at most"). In other words *maximum downtime* is accumulated across

days whereas *reaction time* is mapped to every single invocation of a service.

Coping with the described complexity, our contribution is threefold: We *(i)* provide an in-depth requirements analysis and present related approaches and their shortcomings. Following a design science approach (Hevner, March, Park and Ram, 2004), we *(ii)* develop an ontology framework which enables the generic specification of services, their context and their contractual relations. It facilitates the description of atomic and composite services in an domain-independent fashion. Our approach clusters SLIs into a small finite number of concepts according to their codomain. The aggregation complexity evolving from the potentially infinite number of SLIs and their associated codomains is tremendously reduced and enables automation. Complementary to our ontology framework we design a rule set and aggregation algorithms which enable the automatic processing of service level aggregations incorporating generic classification schemes and process context patterns. To demonstrate the expressive power of our framework we *(iii)* provide an exemplary industrial application scenario and illustrate the functioning and interplay of the designed artifacts.

The remainder of this paper is structured as follows: In Section Requirements & Related Work, requirements upon an approach to automate the creation of service contracts on a process level are identified. Based on these results, Section Automating a "Single Point of Contract" introduces the ontology- and rule-based design of our service contract automation framework that allows for an expressive description of services, agreements, quality characteristics as well as corresponding metrics and aggregation operations on a general abstraction level. To demonstrate the expressive power and applicability of our approach, Section Industrial Application Scenario provides a numerical case study. Finally, Section Conclusion & Future Research summarizes our contribution and outlines open questions and future work.

Requirements & Related Work

In this section, we analyze the fundamental requirements upon a framework for automation of service contracts that allows for the expressive description of technical and composite business services, their quality characteristics, corresponding metrics and aggregation operations. Based on the results, we design a set of artifacts following a design science approach to overcome analyzed shortcomings of existing approaches.

Requirement Analysis

Requirement 1 (Expressive Service Description)**.** *Provide an expressive description of services and complex services, which can be used to semantically describe information about the capability/functionality and quality of a service, which at least includes a description of the information involved in a service invocation (i.e. input and output information), the different possible configurations of a service with the corresponding price (e.g. different service levels with different prices) as well as non-functional quality characteristics, their corresponding metrics and aggregation operations.*

There is already a substantial amount of work in the area of service discovery considering different aspects of a service. Classically a service is considered relevant if it takes the right input types and returns the right output types. Depending on the expressivity of the language in which these types are defined one can distinguish between rather syntactic approaches such as UDDI on the one hand and semantic approaches such as OWL-S, WSMO and SAWSDL on the other hand. The latter uses ontologies to formally describe information required and returned by a service and improve several shortcomings of purely syntactic solutions such as data integration from heterogeneous sources or pure recall and precision.

Since input and output are often not sufficient to completely define the functionality of a service, several extensions have been proposed: As an example, OWL-S, WSMO and several other approaches such as Constantinescu, Binder and Faltings (2005) allow for semantic descriptions of pre- and postconditions/effects, while Berardi, Calvanese, Giacomo, Lenzerini and Mecella (2005); Agarwal (2007) capture the behavior of a service by more detailed description of the service's internal process.

While all the approaches mentioned above do not directly consider non-functional properties of services in the discovery processes, other approaches like Bichler and Kalagnanam (2005); Lamparter, Ankolekar, Grimm and Studer (2007) add expressiveness with respect to quality of service and allow the specification of service levels together with corresponding prices. Thereby, they enable the specification of configurable services. However, these approaches focus only on single services and thus cannot capture all kinds of dependencies between services. Furthermore it remains unclear how quality criterions of a composition of services are specified and computed. In (Blau, van Dinther, Conte, Xu and Weinhardt, 2009) and (Zeng, Benatallah,

Ngu, Dumas, Kalagnanam and Chang, 2004), the problem of aggregating quality aspects along a composition plan is addressed. Nevertheless, only exemplary attribute types such as *throughput* and *availability* are considered. Depending on the attribute types, adequate metrics and aggregation operations are needed in order to measure and compute the overall quality of a service composition.

There are several works that discuss service definitions, aiming at the creation of vocabularies and ontologies for describing services in a general sense (e.g. Ferrario and Guarino (2009); Baida, Gordijn and Omelayenko (2004)). In contrast our approach to service descriptions is complementary to such work, as it does not classify services in different types and treats their instrinsic and general properties. Our work aims at quality attributes and configurations of concrete services, and their aggregation according to their usage context.

Requirement 2 (Service Composition Support)**.** *Support of the composition of complex business services by analyzing service descriptions and representing contextual dependencies that are crucial for the computation of composite quality characteristics. This requires a holistic description of service dependencies in terms of compatibility, quality characteristics and contextual information on service interrelations.*

In the composition step services are composed to a single business service, which might include passing the output of one service through to the input of another service, incorporating technical services, integrating data from different sources, translating data formats and schemas, etc. As finding business services that realize a certain functionality becomes increasingly cumbersome as the number of services grows, machine support for finding valid and suitable compositions is required. In order to tackle this complexity, a wide range of different methods have been proposed for automatically calculating the compositions. Most of these approaches rely on AI planning algorithms and apply backward chaining to derive suitable compositions from a certain goal. Such an approach is, for example, presented in Lécué and Léger (2005) for stateful and in Sirin, Parsia, Wu, Hendler and Nau (2004) for stateless services. While these approaches consider semantic descriptions of input and output, they do not consider dependencies between services. This problem is addressed by composition on process level, where the (temporal) behavior of a service is also considered. Such approaches are presented, e.g., in Berardi, Calvanese, Giacomo, Lenzerini and Mecella (2005).

However, these approaches are not sufficient since they only consider desired functionality as composition goal and disregard quality of service and contextual meta information. Approaches for QoS-based composition are presented in Cardoso, Sheth, Miller, Arnold and Kochut (2004); Zeng, Benatallah, Ngu, Dumas, Kalagnanam et al. (2004); Küster, König-Ries, Klein and Stern (2007). Since they require predefined aggregation rules for all QoS-attributes, they are only partly suitable for full service contract automation in distributed environments.

Other works such as Akkermans, Baida, Gordijn, Peña, Altuna and Laresgoiti (2004) combine services not in terms of data flow, but as service bundles, where services can enhance each other and a combined functionality is provided. The authors generally also treat quality aspects but do not cover how they are integrated in the algorithm for the configuration of service bundles. Certainly the problem of service bundling could also benefit from our approach.

Requirement 3 (Expressive Quality & Contractual Descriptions)**.** *Provide means for an expressive description of QoS-attributes of technical as well as composite business services. Such a description shall comprehend "traditional" quality aspects (e.g. throughput, availability) but also business-oriented indicators (e.g. loss given default, probability of success, degree of target achievement). QoS aspects highly depend on the composition context the service is situated in as well as the corresponding process flow of that composition. Hence, a specification of SLIs is needed that provides metrics and aggregation operations based on the attribute type at hand and the service's situational context.*

As stated by Hill (1977), the value generated by a service is mainly represented by intangible elements exposed at execution. Focus on functional and non-functional properties of a service as well as its context are essential to provide, control and assure QoS. That is, the quality that is experienced by the customer is also directly or indirectly influenced by the service's reputation in certain communities or advertisement campaigns promoting it.

If services are delivered via electronic networks (such as the Web), the provider needs to deal with challenges like unpredictable reliability, low performance of Web protocols, infrastructure problems, and many more. Detailed information on the main aspects of QoS in a Web services context can be found in Cardoso, Sheth, Miller, Arnold and Kochut (2004); Liu, Ngu and Zeng (2004); Mani and Nagarajan (2002); Papazoglou (2008); Zeng, Benatallah, Ngu, Dumas, Kalagnanam et al. (2004). O'Sullivan (2006) goes beyond Web services and quality aspects. He gives an extensive treatment of all kinds of non-functional service properties.

Based on O'Sullivan's work and others, currently the Unified Service Description Language (USDL) is under development.[1] USDL in its current state defines a number of static schema elements for service descriptions, which does not allow for easy extension to different kinds of SLIs, needed in specific applications or domains.

As price competition is tough due to low variable costs of service provisioning, QoS is, from an economic perspective, the key criterion to differentiate from competitors (Papazoglou, 2008).

Academic literature concerning the aggregation of SLIs and SLOs on a business process level can be roughly categorized in three areas whose boundaries blur to some extent. Models which aggregate the SLOs of single SLIs in a mathematical way are introduced in Blake and Cummings (2007); Jaeger, Rojec-Goldmann and Muhl (2004); Unger, Leymann, Mauchart and Scheibler (2008). Models that provide a framework for building a single document out of a set of SLA documents (SLAs of the single services invoked by one business process) are discussed in Ludwig and Franczyk (2008); Muthusamy, Jacobsen, Coulthard, Chan, Waterhouse and Litani (2007); Schmidt (2000). Xiao, Chan, Zou, Benayon, O'Farrell, Litani, Hawkins and Lab (2008); Daly, Kar and Sanders (2002) elaborate models which validate the SLOs on business process level by means of simulations.

Only some of the above mentioned scholars cover the issue of aggregating the SLIs and SLOs on a business process level (Blake and Cummings, 2007; Jaeger, Rojec-Goldmann and Muhl, 2004; Unger, Leymann, Mauchart and Scheibler, 2008). However, their models are limited in a way that they only consider a simplified environment with a small set of SLIs and associated SLOs. In real life scenarios, a huge number of SLIs has to be faced and processed. The semantic support, included in our approach, provides an adequate way to deal with this problem.

Summary & Design Science Contribution

Based on the results from our requirement analysis we intent to cope with the outlined challenges and corresponding shortcomings of existing approaches by designing the following artifacts according to the design science approach firstly introduced by Hevner, March, Park and Ram (2004):

Service Contract Automation Framework – We present an ontology framework which enables the generic specification of services, their context and their contractual relations. It facilitates the description of atomic and composite services in a domain-independent fashion. With respect to a service's context and contractual environment, the framework provides means for the specification of service level agreements and corresponding contractual elements as well as the service's process-related context and its classification into adequate aggregation schemes. In summary, this realizes a full-fledged self-description of services to become autonomic entities – *a service knows everything about itself and its context*.

Generic Classification Rule System – Complementary to our ontology framework we design a rule set which enables the automatic processing of service level aggregations incorporating generic classification schemes and process context patterns. The rule system facilitates the necessary information of the service description and hence autonomically functions without external intervention.

Aggregation Algorithms – Enabling an automated processing of service level aggregation in the context of business processes, we present a generic set of aggregation algorithms that recursively aggregates quality attributes of services within complex business processes. The algorithms facilitate the ontology-based service descriptions and corresponding rule sets as described above in an automatic manner which can be realized in a service-oriented middleware.

AUTOMATING A "SINGLE POINT OF CONTRACT"

This section focuses on the semantic design of the *Service Contract Automation Framework*. We present a two-layered ontology framework that embodies a holistic conceptualization of services and their contractual grounding. It allows for an expressive ontology-based description of services and composite services in general. Furthermore the framework provides means for the in-depth specification of service level agreements and corresponding contractual elements. Before introducing the framework itself we briefly describe the concept of semantics and ontology-based modeling.

[1] `http://www.internet-of-services.com/index.php?id=12`, Accessed April 2010

Ontology Formalism

As a formalism to represent our ontology framework, we use the Web Ontology Language (OWL). OWL is standardized by the World Wide Web Consortium (W3C) (W3C, 2004) and defines with OWL-DL a language for specifying ontologies with formal semantics based on description logics (DL, cf. Baader, Calvanese and McGuinness (2007)). The DL variant used by OWL-DL has the advantage that it is decidable and inference algorithms exist that are more efficient than reasoning with first-order logic. There exist several implementations of OWL-DL inference engines, so-called OWL reasoners.

The logical foundation of OWL facilitates to draw logical conclusions and allows the reasoner to derive implicit knowledge from an ontology. We briefly review some of the modeling constructs of OWL using its DL syntax. The main elements of OWL are *individuals*, *properties* that relate individuals to each other, and *classes* that group together individuals which share some common characteristics. Classes as well as properties can be put into subsumption hierarchies. For example $\texttt{ComplexService} \sqsubseteq \texttt{Service}$ states that every complex service is also a service. If $\texttt{C} \sqsubseteq \texttt{D}$ and $\texttt{D} \sqsubseteq \texttt{C}$ holds, the classes are called equivalent, denoted as $\texttt{C} \equiv \texttt{D}$. Furthermore, OWL allows not only simple named classes but also complex *class constructors* that pose restrictions on the properties of a class. For example, the statement $\texttt{ComplexService} \equiv \exists\ \texttt{hasComponent.Service}$ specifies that complex services are exactly those that have at least one component service.

For the reader's convenience we illustrate our ontology in UML notation (Brockmans, Volz, Eberhart and Loffler, 2004) where UML classes correspond to OWL concepts, UML associations to object properties, UML inheritance to sub-concept relations, UML dependencies to OWL class instantiations and UML attributes to OWL datatype properties.

OWL expressiveness is limited to tree structures. It is for example possible to express that a sequence is the pattern of a complex service that has two component services: one component that has a predecessor and one component that has a successor:

$$\texttt{Sequence} \sqsubseteq \exists \texttt{hasPattern}^{-}.(\texttt{ComplexService} \sqcap \exists \texttt{hasComponent}.(\texttt{Service} \sqcap \exists \texttt{hasSuccessor.Service}) \sqcap \exists \texttt{hasComponent}.(\texttt{Service} \sqcap \exists \texttt{hasPredecessor.Service}))$$

However it is not possible to define that the first component is the predecessor of the second service. In order to model such restrictions we use the DL-safe fragment of SWRL rules (Motik, Sattler and Studer, 2005). Reasoning with OWL-DL and DL-safe rules is still decidable and allows us to model the following sequence definition:

$$\texttt{Sequence}(\texttt{p}) \leftarrow \texttt{hasPattern}(\texttt{c},\texttt{p}) \wedge \texttt{ComplexService}(\texttt{c}) \wedge \texttt{hasComponent}(\texttt{c},\texttt{s1}) \wedge \texttt{hasComponent}(\texttt{c},\texttt{s2}) \wedge \texttt{Service}(\texttt{s1}) \wedge \texttt{hasSuccessor}(\texttt{s1},\texttt{s2}) \wedge \texttt{Service}(\texttt{s2}) \wedge \texttt{hasPredecessor}(\texttt{s2},\texttt{s1}).$$

As DL-safe rules only allow the use of variables that refer to known objects, we implicitly restrict the application of rules only to known instances. In our prototype implementation we use KAON2[2] as reasoner for OWL and rules.

Ontology Framework

We propose an ontology framework for services and agreements which is structured in three parts: The *Service Ontology* – divided into the *Service Context* and the *Service Configuration* – and the *Service Contract Ontology*.

The first part represents the *Service Ontology* which defines relevant concepts to specify services in general – independently from a concrete application scenario (cp. Figure 2).

The *Service Ontology* functions as an ontology design pattern (Devedzic, 1999; Gangemi, 2005) for modeling services and composite services in a standardized manner. The level of abstraction guarantees a detachment from specific application scenarios but also provides concepts and relations which are concrete enough to generate feasible models for services. By the nature of its design, this architecture minimizes maintenance effort as the generic knowledge is strictly separated from domain and scenario specific knowledge. The service ontology is further structured in two parts: *Service Context* and *Service Configuration*.

Concepts in the service context part support the description of contextual information on service interrelations. To provide an approach which can deal with every imaginable process invocation context of a service we decided on implementing the Workflow Patterns from Van Der Aalst, Ter Hofstede, Kiepuszewski and Barros (2003) as the possible process context interrelations. Van Der Aalst, Ter Hofstede, Kiepuszewski and Barros (2003) provide twenty universally valid process patterns designed to evaluate

[2] Available at `http://kaon2.semanticweb.org/`, Accessed April 2010

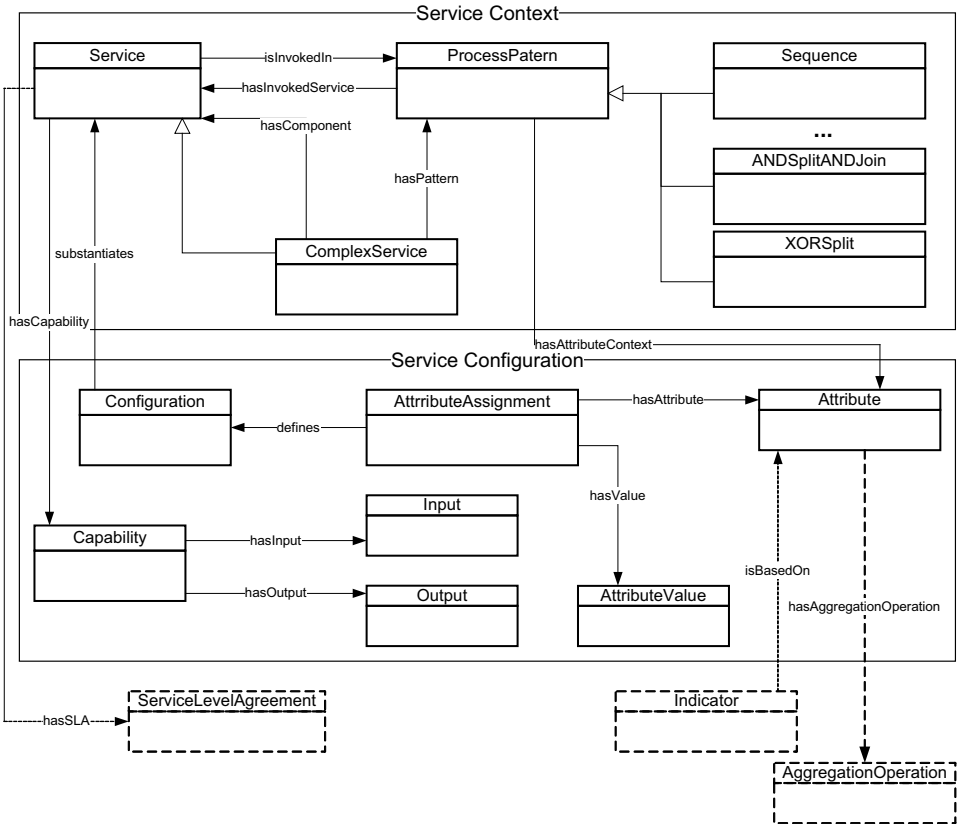

Figure 2: Service Ontology.

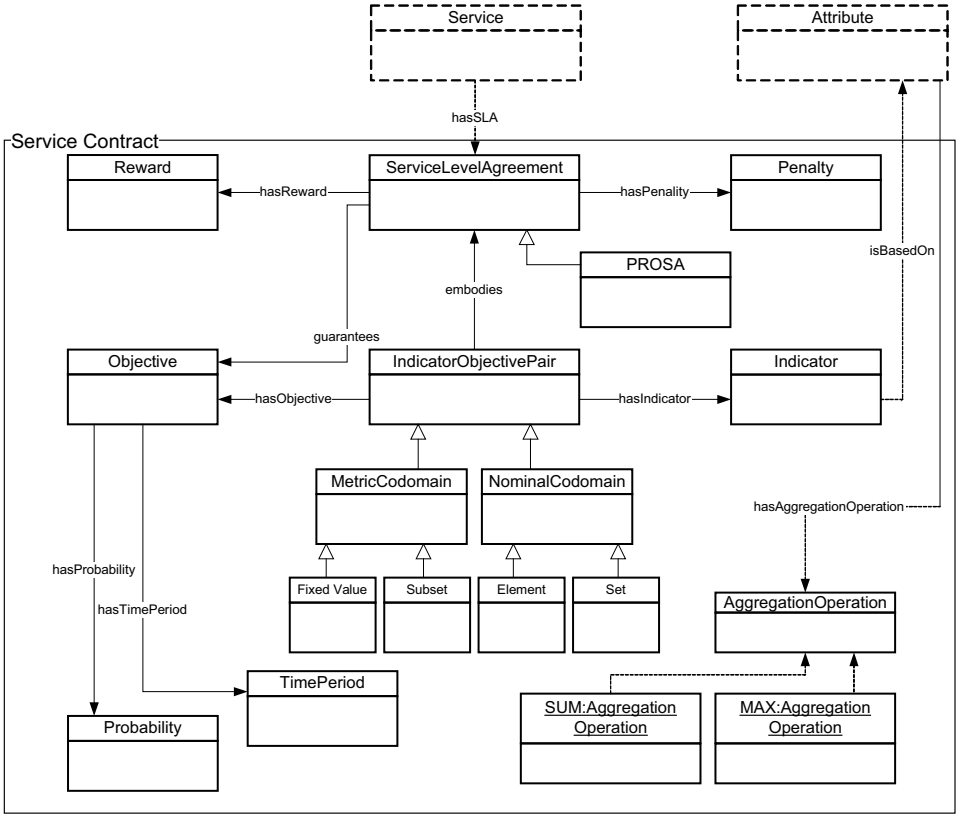

Figure 3: Service Contract Ontology.

the power and expressiveness of Workflow Management Systems (WFMS). Therefore these patterns are of importance during design time as well as during run time. Jaeger, Rojec-Goldmann and Muhl (2004) stated that only 14 of the 20 patterns are relevant during design time in a QoS aggregation context. Using these 14 patterns our approach can handle every design time process constellation. For instance, a service can be invoked in a *sequence*, in a *synchronizing merge* or in an *AND-Split*.

Secondly, the service configuration part represents concepts for describing service capabilities and configurations. A Configuration is defined by an assignment of Attributes and their AttributeValues which form concrete service instances. The attribute concept specifies the semantic type of service property (e.g. *response time*, *encryption mode*). The Capability concept is divided in an Input and Output part which allows for the semantic specification of the service's capabilities.

The Service Contract Ontology conceptualizes service level agreements and corresponding contractual concepts (cp. Figure 3). A ServiceLevelAgreement is specified by one or multiple IndicatorObjectivePairs which again consist of Indicators (SLIs) and corresponding Objectives (SLOs). Indicators are based on one or more service attributes as defined in the service ontology. Depending on the service level indicator and its corresponding objective, the adequate codomain can either be nominal (NominalIndicator, NominalObjective) or metrical (MetricIndicator, MetricObjective). Nominal values can either be single elements or sets whereas metric values can be single fixed values or a subset. For illustration, Table 1 shows different codomains and descriptive examples.

Objectives can further be specified by probabilities and time periods (e.g. Minimum throughput of 10GB with a probability of 80% from Monday to Friday).

Codomain	Granularity	Indicator	Objective
Nominal	Element	Encryption	{True}
Nominal	Set	Encryption type	{SymmetricCypher, AsymmetricCypher}
Metric	Fixed Value	Price	{10}
Metric	Subset	Maximum downtime	[0,10]

Table 1: Exemplary Service Level Indicators and Objectives Depending on their Codomain.

INDUSTRIAL APPLICATION SCENARIO

In order to evaluate the design of the service contract automation framework as presented in Section Automating a "Single Point of Contract", this section provides its application in the context of an industrial scenario. On the one hand, we illustrate the challenges of providing a PROSA that stem from the complexity of a process-oriented service level computation. On the other hand, we demonstrate the expressive power of our framework and its general applicability.

Recall the scenario as depicted in Figure 1. A company implements a business process *customer call* which provides a hotline in order to deal with customer issues and requests. However, an adequate IT infrastructure and services in the context of CRM activities and Workflow Management (WFM) activities are not core competencies of the company. Hence, the board decides on implementing the CRM system and the WFMS from a third-party service provider. (That is, process step *register customer data and customer issue* and process step *delegate issue to responsible colleague* along with the therein invoked services $WFMS_1$, $WFMS_2$, CRM_1 and CRM_2. $WFMS_1$, $WFMS_2$ as well as CRM_1 and CRM_2 are independent virtual applications each.)

To reduce managerial overhead and increase the attractiveness for the customer, the service provider intents to offer an aggregated PROSA (cp. Section Introduction) to the customer which entails the overall quality aspects of the complex service consisting of services $WFMS_1$, $WFMS_2$, CRM_1 and CRM_2 – as depicted in Figure 4. Coping with the complex task of aggregating multiple QoS attributes from possibly dependent single services, the service provider facilitates our framework which allows for the needed service contract automation. For simplicity reasons and the reader's convenience we reduce the scenario to a single QoS attribute *maximum downtime*.

In order to provide an ontology based algorithm for computing the PROSA of the example its necessary to formalize the aggregation operations. To exemplify this fact we provide rules for the operations *sum* and *maximum* as depicted in the following. The rules *Rule SUM* respectively *Rule MAX* define based on our ontology (cp. Section Automating a "Single Point of Contract") whether an attribute of a service has to be aggregated using *sum* or *maximum*. To provide an universally valid ontology beside generally admitted rules and algorithms for aggregation, the examination of other operators is required. In order to reduce complexity of the example only *sum* and *maximum* are considered.

$$\textbf{Rule MAX: } \texttt{hasAggOp(a,MAX)} \leftarrow \texttt{ComplexService(c)} \wedge \texttt{Configuration(cfg)} \wedge \texttt{substantiates(cfg,c)} \wedge \texttt{AttributeAssignment(aa)} \wedge \texttt{defines(aa,cfg)} \wedge \texttt{hasAttribute(aa,a)} \wedge \texttt{MaxDowntime(a)} \wedge \texttt{hasPattern(c,p)} \wedge (\texttt{ANDSplitANDJoin(p)} \vee \texttt{XORSplit(p)})$$

$$\textbf{Rule SUM: } \texttt{hasAggOp(a,SUM)} \leftarrow \texttt{ComplexService(c)} \wedge \texttt{Configuration(cfg)} \wedge \texttt{substantiates(cfg,c)} \wedge \texttt{AttributeAssignment(aa)} \wedge \texttt{defines(aa,cfg)} \wedge \texttt{hasAttribute(aa,a)} \wedge \texttt{MaxDowntime(a)} \wedge \texttt{hasPattern(c,p)} \wedge \texttt{Sequence(p)}$$

Our ontology along with the introduced rules build the base for implementing the aggregation algorithm which is described in the following. Executing this algorithm iteratively for each workflow pattern in the process enables the overall QoS aggregation for every imaginable process context. The composition of services invoked in the same process pattern is regarded as a complex service.

We complete this section by aggregating the above mentioned PROSA out of the energy domain with a numerical example. Figure 5 illustrates the above described scenario. See Appendix 1 for an exposition of the execution. This demonstrates one of

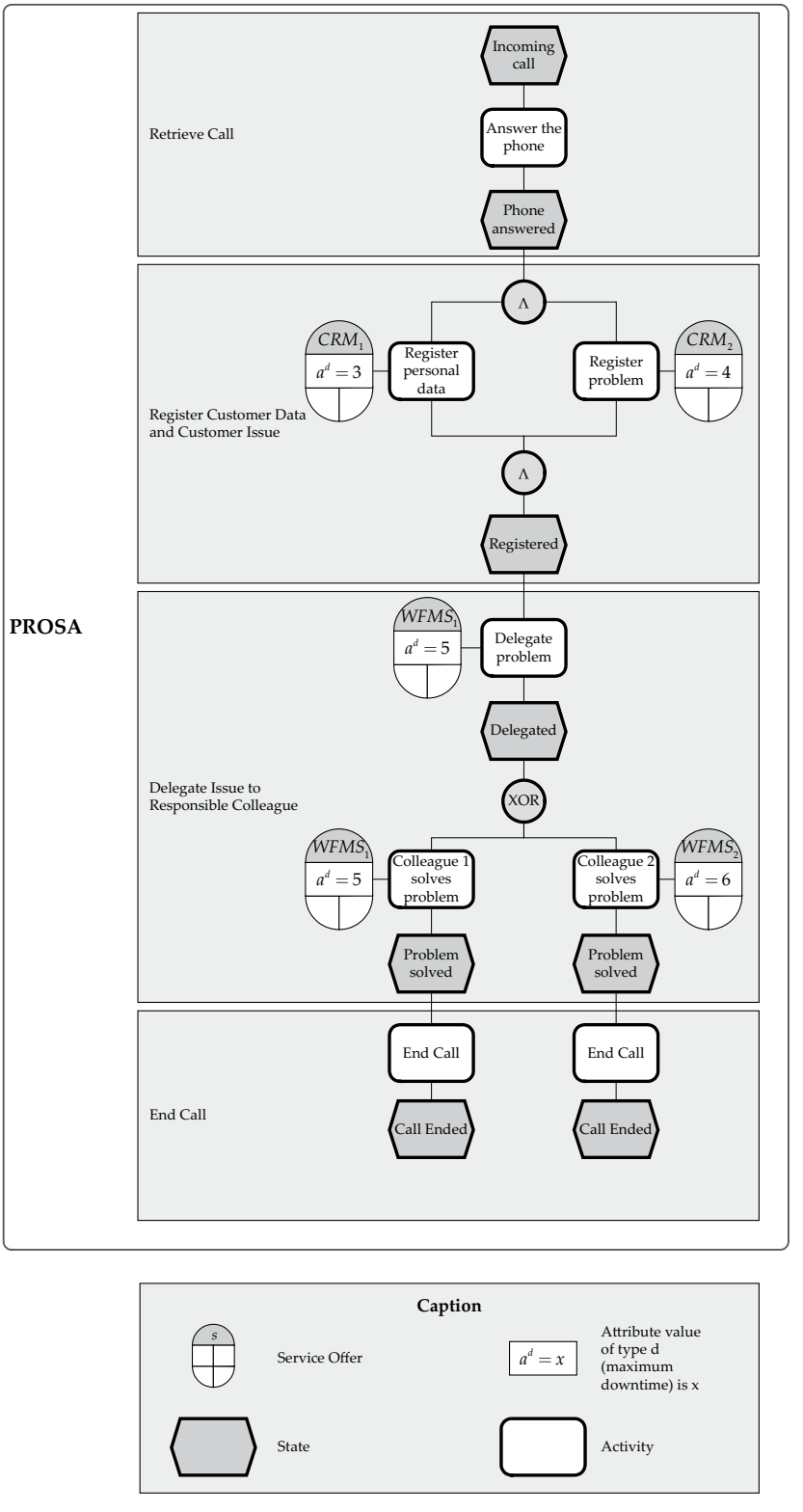

Figure 4: Process-Oriented Service Level Agreement for the Requested CRM Services.

Algorithm 1 computeOverallQuality

```
Require: c, a
q = 0
if ComplexService(c) then
  o = hasAggOp⁻.{a}
  for all s ∈ ∃hasComponent⁻.{c} do
    if ComplexService(s) then
      q = o(q, computeOverallQuality(s,a))
    else
      v = getValue(s,a)
      q = o(v,q)
    end if
  end for
else
  q = getValue(c,a)
end if
return q
```

the main strengths of our algorithm – recursion to deal with every imaginable service composition scenario independently from the instance level.

CONCLUSION & FUTURE RESEARCH

The current trend of compensating a lack of core competencies by integrating external services from specialized providers into internal end-to-end business processes demands for a comprehensive management of SLAs. From a provider perspective, the task of computing the overall QoS criterions of the business service which is requested from the customer is a crucial procedure in order to reduce managerial overhead and to reduce the risk of penalties through non-performance. Depending on the type of quality attribute and the composition context the service is invoked in, the aggregation of multiple QoS aspects into a process-oriented service level agreement (PROSA) requires different aggregation operations (mathematical as well as semantic- and rule-based). As the time horizon of SLAs is constantly increasing in the context of Web services provisioning – in an on-demand fashion – the complex task of service contract (PROSA) creation requires an automation approach.

Addressing these challenges, we provided a holistic framework for the automation of contract creation for complex services on a process level. It allows for an expressive description of services, agreements, quality and corresponding aggregation operations on a general level in order to assure its applicability in different scenarios. Our approach is based on an ontology-based classification schema which clusters indicators according to their codomain. The aggregation complexity evolving from the potentially infinite number of indicators and their associated codomains is tremendously reduced and consequently automation is enabled. A set of algorithms leverages the generality of the service description and fosters an automated creation of PROSAs. To demonstrate the expressive power of our framework we additionally provided an industrial application scenario.

As future work, we intent to extend our ontologies in order to capture indicators and objectives on a higher level of abstraction, i.e. the representation of business-oriented Key Performance Indicators (KPIs) which have managerial implications and provide support for strategic recommendations. Such KPIs are based on multiple indicators and represent e.g. *processing time*, *process cost rate* or *degree of capacity utilization*. As the presented methodology appears to rise costs, one of our next steps is to provide an argumentative analysis of benefits and costs. Not only in the context of the cost analysis but also in order to further support the relevancy of our work, we are currently working on additional case studies e.g. in the energy domain. In this context we plan to provide an implementation that is built upon state of the art technical and business process monitoring applications.

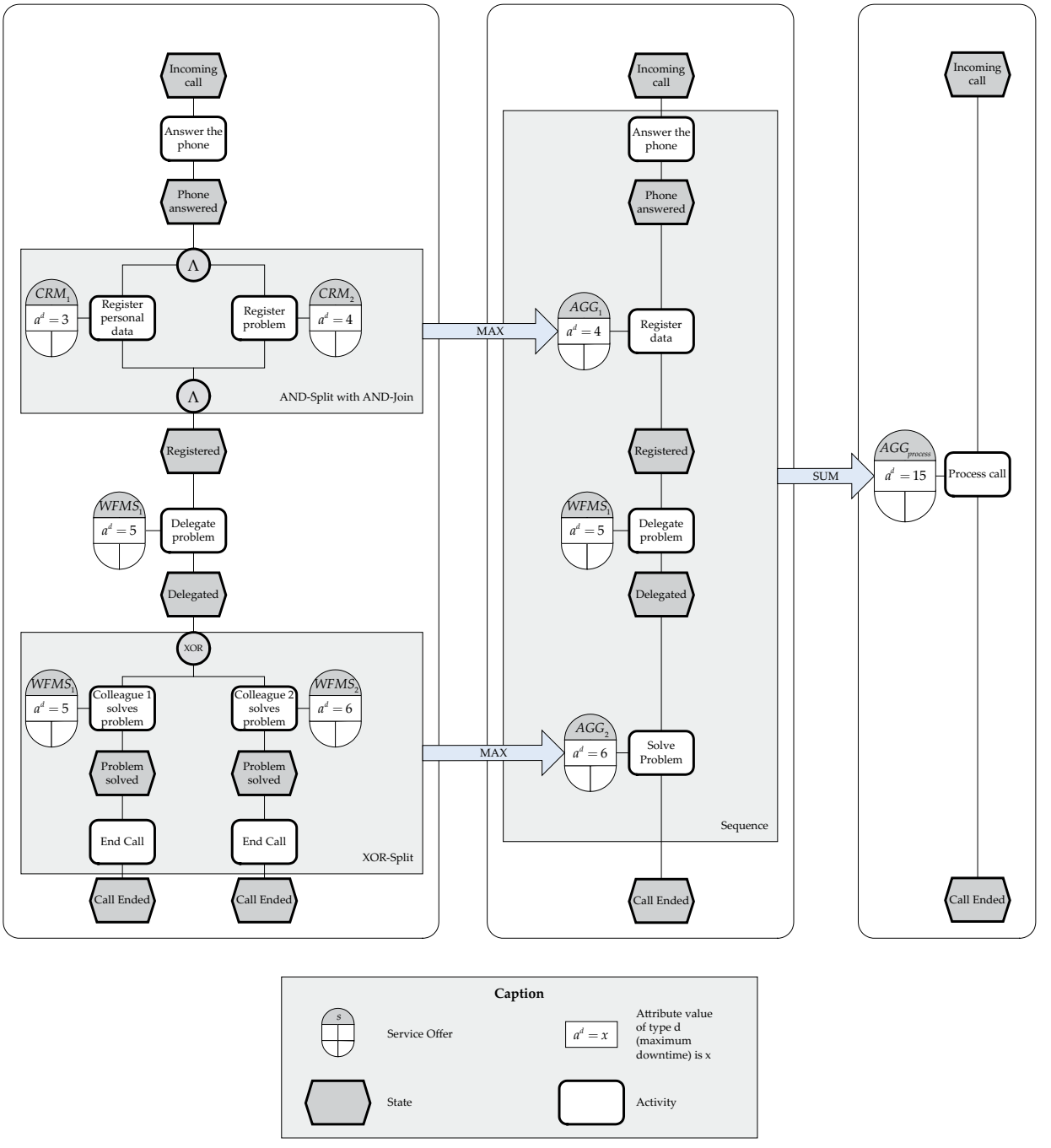

Figure 5: Numerical Example for a Process-Oriented Service Level Agreement Involving CRM and WFM Services.

REFERENCES

1. Agarwal, S. (2007) *Formal Description of Web Services for Expressive Matchmaking*, Ph.D. thesis, Fakultät für Wirtschaftswissenschaften, Universität Karlsruhe (TH).
2. Akkermans, H., Baida, Z., Gordijn, J., Peña, N., Altuna, A. and Laresgoiti, I. n. (2004) Value Webs : Using Ontologies to Bundle Real-World Services, *IEEE Intelligent Systems*, , July/August.
3. Baader, F., Calvanese, D. and McGuinness, D. L. (2007) *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press.
4. Baida, Z., Gordijn, J. and Omelayenko, B. (2004) A shared service terminology for online service provisioning, in *Proceedings of the 6th international conference on Electronic commerce (ICEC '04)*, ACM Press, New York, New York, USA.
5. Berardi, D., Calvanese, D., Giacomo, G. D., Lenzerini, M. and Mecella, M. (2005) Automatic Services Composition based on Behavioral Descriptions, *Int. J. of Cooperative Information Systems (IJCIS)*, 14, 4, 333376.
6. Bichler, M. and Kalagnanam, J. (2005) Configurable Offers and Winner Determination in Multi-attribute Auctions, *European Journal of Operational Research*, 160, 2, 380–394.
7. Blake, M. and Cummings, D. (2007) Workflow composition of service level agreements, in *Proceedings of the IEEE International Conference on Services Computing*, 138–145.
8. Blau, B., van Dinther, C., Conte, T., Xu, Y. and Weinhardt, C. (2009) How to Coordinate Value Generation in Service Networks? - A Mechanism Design Approach, *Journal of Business and Information Systems Engineering (Wirtschaftsinformatik)*, 1, 5, 343–356, iSSN: 1867-0202.
9. Brockmans, S., Volz, R., Eberhart, A. and Loffler, P. (2004) Visual Modeling of OWL DL Ontologies Using UML, *The Semantic Web–ISWC*, 198–213.
10. Cardoso, J., Sheth, A., Miller, J. A., Arnold, J. and Kochut, K. (2004) Quality of Service for Workflows and Web Service Processes, *Journal of Web Semantics*, 1, 3, 281–308.
11. Constantinescu, I., Binder, W. and Faltings, B. (2005) Flexible and Efficient Matchmaking and Ranking in Service Directories, in *IEEE Int. Conf. on Web Services (ISWC'05)*, Orlando, USA, 5–12.
12. Daly, D., Kar, G. and Sanders, W. (2002) Modeling of service-level agreements for composed services, *Lecture notes in computer science*, 4–15.
13. Devedzic, V. (1999) Ontologies: Borrowing from Software Patterns, *Intelligence*, 10, 3, 14–24.
14. Ferrario, R. and Guarino, N. (2009) Towards an Ontological Foundation for Services Science, in *FIS 2008*, 152–169.
15. Gangemi, A. (2005) Ontology Design Patterns for Semantic Web Content, *Proceedings ISWC 2005*, 262–276.
16. Hevner, A., March, S., Park, J. and Ram, S. (2004) Design Science in Information Systems Research, *Management Information Systems Quarterly*, 28, 1, 75–106.
17. Hill, T. (1977) On Goods and Services, *Review of Income and Wealth*, 23, 4, 315–338.
18. Jaeger, M., Rojec-Goldmann, G. and Muhl, G. (2004) QoS aggregation for Web service composition using workflow patterns, in *Proceedings of the 8th Enterprise Distributed Object Computing Conference*, volume 4, Citeseer, 149–159.
19. Jin, L., Machiraju, V. and Sahai, A. (2002) Analysis on service level agreement of web services, Technical report.
20. Knapper, R., Conte, T., Blau, B. and Richter, H. (2010) PROSA – Process-Oriented Service Level Agreements for Providing a Single Point of Contract, in *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI)*, Goettingen.

21. Küster, U., König-Ries, B., Klein, M. and Stern, M. (2007) DIANE - A Matchmaking-Centered Framework for Automated Service Discovery, Composition, Binding and Invocation, in *Proc. of 16th Int. WWW 2007*, Banff, Canada.

22. Lamparter, S., Ankolekar, A., Grimm, S. and Studer, R. (2007) Preference-based Selection of Highly Configurable Web Services, in *Proceedings of the 16th International World Wide Web Conference*, Banff, Canada, 1013–1022.

23. Lécué, F. and Léger, A. (2005) A Formal Model for Semantic Web Service Composition, in *ISWC the 5th International Semantic Web Conference*, 385–398.

24. Liu, Y., Ngu, A. and Zeng, L. (2004) QoS computation and policing in dynamic web service selection, in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, ACM New York, NY, USA, 66–73.

25. Ludwig, A. and Franczyk, B. (2008) COSMA—An Approach for Managing SLAs in Composite Services, in *Proceedings of the 6th International Conference on Service-Oriented Computing*, Springer, 632.

26. Mani, A. and Nagarajan, A. (2002) Understanding quality of service for Web services, *IBM developerWorks*.

27. Motik, B., Sattler, U. and Studer, R. (2005) Query Answering for OWL-DL with Rules, *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3, 1, 41–60.

28. Muthusamy, V., Jacobsen, H., Coulthard, P., Chan, A., Waterhouse, J. and Litani, E. (2007) SLA-driven business process management in SOA, in *Proceedings of the 2007 conference of the center for advanced studies on Collaborative research*, ACM, 267.

29. O'Sullivan, J. (2006) *Towards a Precise Understanding of Service Properties*, Ph.D. thesis, Queensland University of Technology.

30. Papazoglou, P. (2008) *Web Services: Principles and Technologies*, Prentice Hall.

31. Schmidt, H. (2000) Service Level Agreements based on Business Process Modeling, in *HP OpenView University Association Workshop*, Citeseer.

32. Sirin, E., Parsia, B., Wu, D., Hendler, J. and Nau, D. S. (2004) HTN planning for web service composition using SHOP2, *Journal of Web Semantics*, 1, 4, 377396.

33. Unger, T., Leymann, F., Mauchart, S. and Scheibler, T. (2008) Aggregation of service level agreements in the context of business processes, in *Proceedings of the 12th Enterprise Distributed Object Computing Conference*, 43–52.

34. Van Der Aalst, W., Ter Hofstede, A., Kiepuszewski, B. and Barros, A. (2003) Workflow patterns, *Distributed and parallel databases*, 14, 1, 5–51.

35. W3C (2004) Web Ontology Language (OWL), `http://www.w3.org/2004/OWL`.

36. Xiao, H., Chan, B., Zou, Y., Benayon, J., O'Farrell, B., Litani, E., Hawkins, J. and Lab, I. (2008) A Framework for Verifying SLA Compliance in Composed Services, in *Proceedings of the 2008 IEEE International Conference on Web Services*, IEEE Computer Society, 457–464.

37. Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J. and Chang, H. (2004) QoS-Aware Middleware for Web Services Composition, *IEEE Transactions on Software Engineering*, 30, 5, 311–327.

APPENDIX

NUMERICAL EXAMPLE (STEP-WISE)

Require: $AGG_{process}$, MaximumDowntime
1: q = 0
2: ComplexService($AGG_{process}$)
3: o = SUM
4: hasComponent($AGG_{process}$, AGG_1) $\wedge$ ComplexService(AGG_1)
5: q' = 0
6: o' = MAX
7: hasComponent(AGG_1, CRM_1) $\wedge$ $\neg$ComplexService(CRM_1)
8: q'' = 3
9: q' = MAX(q',q'') = 3
10: hasComponent(AGG_1, CRM_2) $\wedge$ $\neg$ComplexService(CRM_2)
11: q'' = 4
12: q' = MAX(q',q'') = 4
13: q = SUM(q,q') = 4
14: hasComponent($AGG_{process}$, $WFMS_1$) $\wedge$ $\neg$ComplexService($WFMS_1$)
15: q' = 5
16: q = SUM(q,q') = 9
17: hasComponent($AGG_{process}$, AGG_2) $\wedge$ ComplexService(AGG_2)
18: q' = 0
19: o' = MAX
20: hasComponent(AGG_2,$WFMS_1$) $\wedge$ $\neg$ComplexService($WFMS_1$)
21: q'' = 5
22: q' = MAX(q',q'') = 5
23: hasComponent(AGG_2,$WFMS_2$) $\wedge$ $\neg$ComplexService($WFMS_2$)
24: q'' = 6
25: q' = MAX(q',q'') = 6
26: q = SUM(q,q') = 15
27: **return** 15

SUPPORTING ALGORITHMS

Algorithm 2 getValue

Require: s, a
1: **for all** aa $\in$ $\exists$defines$^-$.($\exists$substantiates$^-$.{s}) **do**
2: **if** aa hasAttribute a **then**
3: v = hasValue$^-$.{aa}
4: **end if**
5: **end for**
6: **return** v

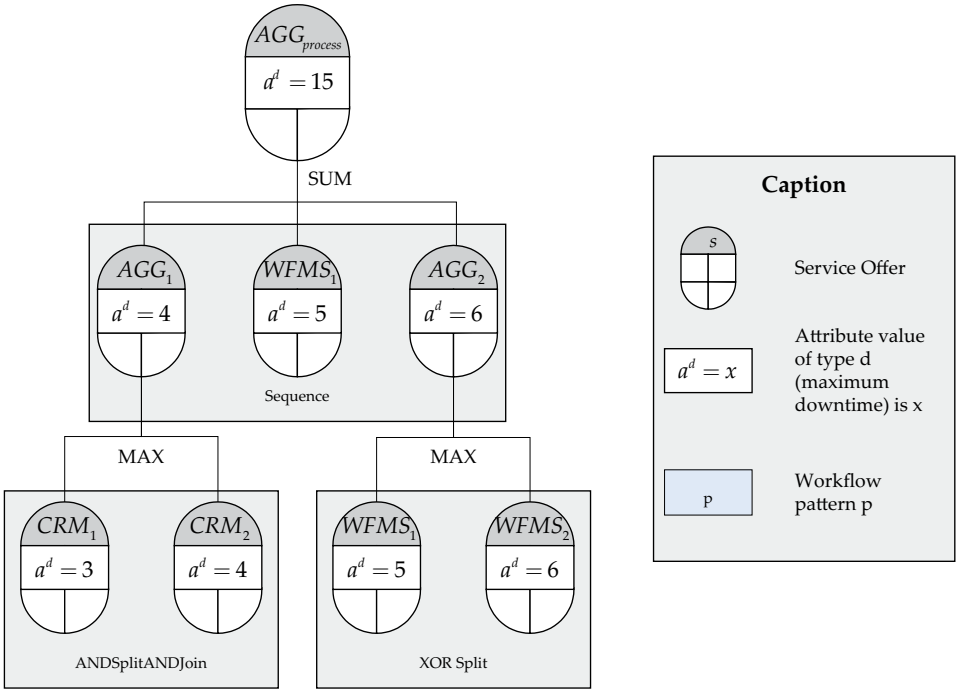

Figure 6: Graphic Representation of the Numerical Example.