Linked USDL Extension for Describing Business Services and Users' Requirements in a Cloud Context

Hind Benfenatki, Catarina Ferreira da Silva, Aïcha-Nabila Benharkat, Parisa Ghodous, Zakaria Maamar

This is a postprint version. The final version is available at DOI: 10.4018/IJSSOE.2017070102
2017, International Journal of Systems and Service-Oriented Engineering

Linked USDL Extension for Describing Business Services and Users' Requirements in a Cloud Context

Hind Benfenatki[1], Catarina Ferreira Da Silva[1], Aïcha-Nabila Benharkat[3]
Parisa Ghodous[1], and Zakaria Maamar[4]

[1]Univ Lyon, Université Claude Bernard Lyon 1, LIRIS UMR 5205 CNRS, 69621 Villeurbanne Cedex, France
[3]LIRIS, CNRS, UMR 5205, INSA - Lyon, 69621 Lyon, France
[4]Zayed University, Dubai, UAE

ABSTRACT

Linked Unified Service Description Language (Linked USDL) provides a comprehensive way for describing services from operational, technical, and business perspectives. However, this description treats services as isolated components that offer functionalities only without emphasis on how they are used. This paper discusses how to extend Linked USDL in a way that permits to describe the services of a marketplace in support of automating the provisioning of service-oriented cloud-based business applications along with satisfying users' requirements. The marketplace consists of business services that can be composed and specialized services that act on behalf of the infrastructure upon which these applications are deployed. A set of experiments demonstrating the use of the extended Linked USDL are also presented in the paper.

Keywords: Linked USDL, service description, service composition, marketplace.

1 INTRODUCTION

There is a persistent trend of developing business applications using a set of loosely-coupled services that are selected with respect to their functionalities and then put together in response to specific users' requirements. Web services usually exemplify these services although other types of services exist such as data services (Lagares Lemos, Daniel, & Benatallah, 2015), user interface services (Lagares Lemos, Daniel, & Benatallah, 2015), human services (Services), and business services (Lüftenegger, 2014). Service classification also exists based on their branches of activities, for instance transport and telecommunications (International classification of goods and services for the purposes of the registration of marks, 2001), tangibility (Bhasin, 2016), and deployability. In this paper, we target business services that are software packages providing business functionalities and subject to composition. For instance, Wiki engines and databases could be composed together in response to a request of making the Wiki content stored persistently.

Service description and composition are widely discussed at the operational level with focus on Input/Output (I/O) matching. However, I/O matching cannot be used to evaluate the composability of business services. Some languages like Linked Unified Service Description Language (USDL) (Cardoso & Pedrinaci, 2015) (Linked USDL, 2015) allow to describe (non-business) services at the business (e.g., price specification), technical (e.g., message protocol), and operational (e.g., service functionalities) levels. However, (i) these languages do not offer any support to business services composition which is quite different from Web services composition (they are not characterized by their I/O), and (ii) they describe what the services do with little regard to how they are used, i.e., environment in which they are deployed, with whom they can be used, etc.

To address the aforementioned two limitations, we build upon our MADONA project standing for Method for AutomateD provisioning (composition and deployment of services) of clOud-based service-oriented busiNess Applications (Benfenatki et al., 2016) to automate the provisioning of business service-oriented applications on cloud environment. Because composability of business services cannot be evaluated using matching of services' input/outputs, we consider service description languages that would allow to identify each service's composition constraints and possibilities (the services with which the described service can be composed). In this paper, we analyse existing languages describing services and user requirements. We target Linked USDL for the description of services and user's requirements. The choice of Linked USDL is guided by the fact that the latter allows a wide coverage of technical, business, and functional aspects when describing services. Moreover, it allows the description of various services for instance cloud, Web, and business services.

Existing user requirements description languages like Web Service Request Language (WSRL) (Mitra, Zhou, Bouguettaya, & Liu, 2013) and Service Requirement Modelling Ontology (SRMO) (Xiang, Liu, Qiao, & Yang, 2007) require that users are familiar with the underlying language's notation, which is quite impossible for non-tech savvy users. Furthermore, these languages describe the requirements in the form of a control flow and/or a data flow. This requires a good knowledge of the business process of the future application to develop, which is quite impossible too when targeting such users. In this paper, we discuss how the necessary support is provided to users by extending Linked USDL.

The rest of this paper is organized as follows. Section 2 describes existing service and user's requirements description languages. Sections 3 and 4 present respectively an overview of Linked USDL and of its extension. Section 5 evaluates our work. Section 6 draws final conclusions and perspectives.

2 RELATED WORK

This section presents 2 categories of languages for describing services and user requirements, respectively.

2.1 Service description languages

Our literature review resulted into classifying service description languages into two categories: (1) those that treat services as isolated components (Christensen, Curbera, Meredith, & Weerawarana, 2001), and (2) those that focus on relationships between services (Cardoso, 2013). Relationships could be either established or potential. The former describes current or past service's composition forming for instance, a service offering. The latter describes with whom a service can or must be composed. Table 1 classifies these works categories of languages using 5 criteria: C1: type of service, C2: technical description, C3: semantic description, C4: description of Quality of Services (QoS), and C5: description of relationships of a service. These 5 criteria provide an exhaustive coverage of the issues to address, namely limited support to business services composition and limited description of how business services are used.

Out of Table 1, Linked USDL has a wide coverage of service types such as business, cloud, and Web. Furthermore, Linked USDL allows to describe various aspects including business, technical, and operational. Moreover, Linked USDL is based on Linked Data (Linked data, 2015) principles, so adding new concepts can extend it. We have therefore used this language to describe services. However, Linked USDL does not describe potential relationships of a service. Instead, it describes the established relationships between services constituting a service offering.

Table 1. Classification of service description languages according to the type of the modelled description

Research work	C1	C2	C3	C4	C5
Web Service Description Language (WSDL) (Christensen, Curbera, Meredith, & Weerawarana, 2001)	Web services	+[1]			
QoS for WSDL (D'Ambrogio, 2006)	Web services	+		+	
(Becha & Amyot, 2014)	Web services			+	
OWL-S (Martin et al., 2004)	Web services	+	+		
SAWSDL (Semantic annotations for WSDL and XML schema, 2007)	Web services	+	+	Can be done by the instantiation of an ontology describing the QoS	
(Afify, Moawad, Badr, & Tolba, 2014)	Software as a Service (SaaS)	+	+	+	
(Taekgyeong & Sim, 2010)	Cloud services	+	+		
(Kan & Sim, 2011)	Cloud services	+	+		
(Tahamtan, Beheshti, Anjomshoaa, & Tjoa, 2012)	Cloud services	+	+		
Cloud# (Liu & Zic, 2011)	Cloud services	+			
BDL (Taher, Nguyen, Lelli, Heuvel, & Papazoglou, 2012)	Cloud services	+		+	
SaaS DL (Sun, Zhang, Chen, Zhang, & Liang, 2007)	SaaS	+			
CSMIC (Cloud service measurement index consortium)	Cloud services			+	
WSMO (Web service modeling ontology (wsmo), 2008)	Web services	+	+	+	+
LinkedWS (Maamar et al., 2011)	Web services				+
Service network (Cardoso, 2013)	Web services		+	+	+
Blueprint (Nguyen, Lelli, Papazoglou, & Heuvel, 2012)	Cloud services	+	+	+	+
CoCoon (Zhang et al., 2012)	IaaS Services	+	+	+	+
Linked USDL (Linked USDL, 2015) (Cardoso &	Business services,	+	+	+	+

[1] Criterion met.

Pedrinaci, 2015)	software services, cloud services, infrastructure services, and human services				

We depict in the following how the literature examines relationships of a service:

1. Association (Cardoso, 2013) and collaboration (Maamar et al., 2011) describe the established relationships of services with respect to previous compositions. The knowledge of these relationships is not exhaustive since it is built up as compositions occur. Moreover, these relationship descriptions do not distinguish between the services that have to be composed (composition constraints) and those that can be composed (composition possibilities). This does not allow to know the minimum necessary composition allowing the proper functioning of a given business service. In our work, we have therefore chosen to describe both composition constraints and composition possibilities of each business service.
2. Complementary and competitor relationships (Cardoso, 2013). Complementarity implies that it is possible to compose services. However, identifying a service's complement relies on user's evaluation of service relationships. This does not allow to describe all the composition possibilities of a service.
3. Service requirements defined by Ngyuyen et al., (Nguyen, Lelli, Papazoglou, & Heuvel, 2012) include in a single concept environment, resource, and service composition constraints. This does not allow to automate neither the composition nor the deployment of services. In fact, it is not possible to automatically identify the requirement type that is represented (composition or deployment constraint). We have therefore distinguished in the description of each service between the concepts describing the composition constraints, environment constraints, and resources constraints.
4. Service offerings (Cardoso & Pedrinaci, 2015) refer to combining services (composition relationships). In the literature, there are works that describe established relationships in service offerings (Cardoso & Pedrinaci, 2015), (Web service modeling ontology (wsmo), 2008), (Maamar et al., 2011), (Cardoso, 2013), (Zhang, Ranjan, Haller, Georgakopoulos, Menzel, & Nepal, 2012) and those describing potential relationships (Nguyen, Lelli, Papazoglou, & Heuvel, 2012), (Zhang et al., 2012). Established relationships do not describe all service's compositions. Potential relationships described in (Zhang et al., 2012) are specific to Infrastructure as a Service (IaaS) services, linking storage to computing services, and network to computing services. This does not allow to describe composition constraints and possibilities.

Table 2. Classification of service description languages according to the type of described relationship

Research work	**C1**	**C2**	**C3**
WSMO (Web service modeling ontology (wsmo), 2008)	+		
LinkedWS (Maamar et al., 2011)	+		
Linked services (Cardoso, 2013)	+		
Blueprint (Nguyen, Lelli, Papazoglou, & Heuvel, 2012)			+
CoCoon (Zhang et al., 2012)	+	+	
Linked USDL (Cardoso & Pedrinaci, 2015)	+		

Table 2 classifies the works describing the relationships of a service based on the following classification criteria: C1: describes *established* relationships between services, C2: describes *potential* relationships between services, and C3: describes *potential constraint* relationships between services.

Unfortunately, not a single work allows to simultaneously describe a service's composition possibilities and constraints. We describe in this work for each service its composition constraints and possibilities. We also define the concepts of environment, and resources constraints. To automate the configuration of business applications on cloud environments, we also describe the configurable parameters of each business service.

2.2 User Requirement Elicitation

Automating the management of cloud services greatly reduce the technical knowledge required for their use. In our previous work (Benfenatki et al., 2016), we automated the generation of service-oriented cloud applications based on non-technical user requirements expressed via a Web form. In order to select the requirements to consider while provisioning service-oriented business applications we have chosen the following criteria to compare in Table 3 the requirements description languages: C1: objective of the work and C2: requirement description abstraction-level. This describes the level of necessary details for the described requirements.

Table 3. Requirement description works comparison

Research works	C1: Objective of the work	C2: abstraction level
BPMN (Business Process Model and Notation, 2011)	Business process description	Desired activities, expected events, control flow, several collaboration participants, message flow
WS-BPEL (Alves et al., 2007)	Description of service-oriented business process execution	Control flow, data flow
WS-CDL (WS-CDL, 2005)	Peer to peer choreography description	Control flow, data flow
SRMO (Xiang, Liu, Qiao, & Yang, 2007)	Requirements description ontology	Desired functionalities, functionalities relationships, quality attributes, control flow
(Yuan & Zhang, 2015) for the development of service-oriented Product Line Software (PLS)	Allows the identification of the specific requirements of a particular client	Desired functionalities, functioning environment, constraints associated to each functionality, requirement rank, requirements relationships (contradiction, quality, etc)
WSRL (Mitra, Zhou, Bouguettaya, & Liu, 2013)	Declarative description of service-oriented request	Desired functionalities, I/O, WSRL request

Works in Table 3 describe the requirements on service-oriented applications. They define a language, an ontology, or a notation for the description of the requirements. These works require from the user to know the defined languages so that she uses them. They can be distinguished by the abstraction level when describing application requirements (C2). In fact, the description of a control flow and/or a data flow requires a good knowledge of the business process to develop. Furthermore, this is not favourable when using business services because they are not defined by their Inputs/Outputs; business services have a higher abstraction level than Web services, and their composition does not consist in invoking service operations, but rather to adapt a business service so that it can cooperate with another.

3 LINKED UNIFIED SERVICE DESCRIPTION LANGUAGE OVERVIEW

USDL (Cardoso, Barros, May, & Kylau, 2010) (USDL, 2011) describes technical and business services to allow services to become usable. Attensity and SAP Research among others initiated this language that was submitted for standardization to the W3C. The description of services with USDL is

based on three perspectives: business (covers, among other things, quality of service, pricing models, and legal constraints), operational (concerns the operations of a service and its functionalities), and technical (includes transport, messages, metadata exchanges, and security protocols).

Linked USDL (Linked USDL, 2015), (Cardoso & Pedrinaci, 2015), (Linked USDL modules, 2015) is a semantic language based on USDL. It describes human services (e.g., consulting), business services (e.g., purchase requisitions), software services (e.g., RESTful services), infrastructure services (e.g., CPU and storage services), etc. The objective of Linked USDL is to allow an open, adaptable, and extensible description of services using decentralized management.

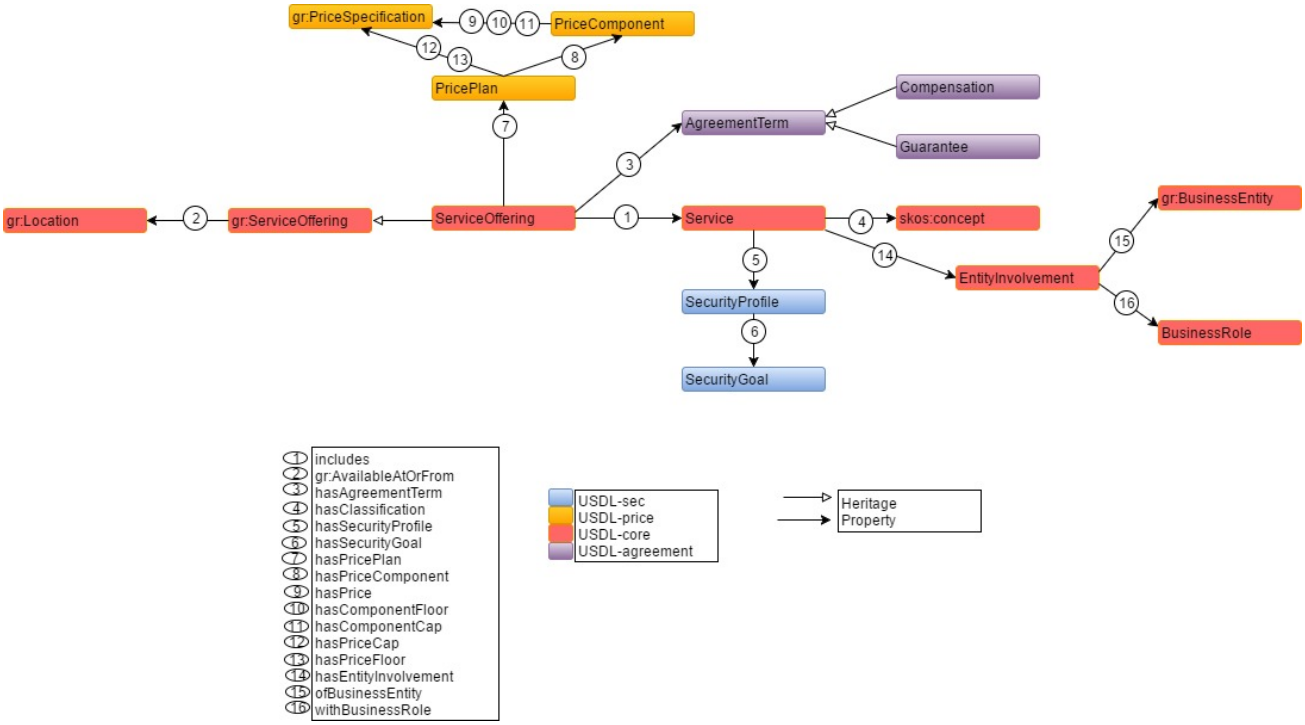

Figure 1. Macroscopic view of Linked USDL

Linked USDL is divided into five modules that have different levels of maturity. Each module is a set of concepts and properties. We illustrate in Figure 1 some Linked USDL classes. A detailed description of all modules is available at (Linked USDL modules, 2015). The five modules are the following:

- **USDL-core**: describes the operational aspects of a service.
- **USDL-price**: describes the price structure of a service.
- **USDL agreement**: describes the quality of the service provided, such as response time and availability.
- **USDL-sec**: describes the security properties of a service.
- **USDL-ipr**: describes the rights to use a service.

Despite Linked USDL advantages such as extensibility and coverage of business, operational, and technical aspects, it does not capture relationships among services. In fact, the type of service relationships described in Linked USDL is done with the class "usdl-core:ServiceOffering". The latter describes the combined services constituting a service offering. However, all possible compositions are not necessarily included in a service offering, since the latter is supplier dependent. Furthermore, Linked USDL describes services as isolated components, hence it has limitations regarding the description of the whole service-oriented application requirements. To overcome these shortcomings, we extend Linked USDL to describe a marketplace of services and requirements of users.

4 LINKED USDL EXTENSION

This section is composed of two parts describing Linked USDL extension. The first one concerns service description. The second one concerns the user's requirements description. In the following we assume the existence of a marketplace of services (business and infrastructure). Each business service has deployment and configuration scripts, and an additional script that connects it to other business services. Juju store is an example of marketplace (Juju Charms, 2016). Service description using Linked USDL can be formatted using RDF (RDF, 2014), Turtle (Turtle recommandation, 2014), or JSON. In our case, we chose to use the turtle format because it allows a simple, concise, and human understandable description. Therefore, we are focusing in this paper on the definition of the new concepts of the extended Linked USDL instead of defining the syntax of the extended language.

4.1 Extending Linked USDL for service description

In Section 2, we analysed the works on a service's established and potential relationships. None of those works allows to describe both composition constraints and possibilities exhaustively. Figure 2 illustrates the extended Linked USDL. We describe in the following the concepts that we have defined to extend the model.

The extension of Linked USDL consists of describing the composition relationships of a business service including its deployment constraints, configurable parameters, category, deployment state, technical characteristics of an IaaS (for business services deployment, only), and QoS description of a business and IaaS services. In the following, each new concept is motivated, defined, and exemplified with an example.

4.1.1 Composition relationships

A business service S1 can be linked to S2 thus creating a composition relationship. The latter includes the composition constraints and possibilities (Definitions 1 and 2).

Composition constraints

Some services need other services to function properly.

Definition 1. Composition constraints link the described business service to another. Composition constraints are either hard or soft.

- Hard constraints (property 11 of Figure 2) impose the services that must be composed to the one described. For example, MediaWiki must be composed with a MySQL database.
- Soft constraints (property 12 in Figure 2) offer the choice of selecting one, and only one, service in a business service family providing the same functionality. For example, Joomla can be composed with a MSSQL, PostgreSQL, or MySQL database.

Composition possibilities

Service-Oriented Applications (SOA) relies on I/O matching during service composition. However, this does not apply to business services. Section 2 has shown the lack of representation of composition possibilities. In fact, studied work described the different past compositions of a service (Maamar et al., 2011) (Cardoso, 2013). The latter do not distinguish between composition constraints and possibilities neither allow an exhaustive description of the potential relationships of a service. To overcome these shortcomings, we describe for each business service its composition possibilities.

Definition 2. Composition possibilities (property 13 of Figure 2) bind the described business service to other peers. A business service S1 is a composition possibility of S2 if and only if S1 can be composed with S2 and S2 works correctly if it is not composed with S1. For example, OpenStack

works properly without a dashboard, but "Horizon" dashboard represents a composition possibility of the core components of OpenStack.

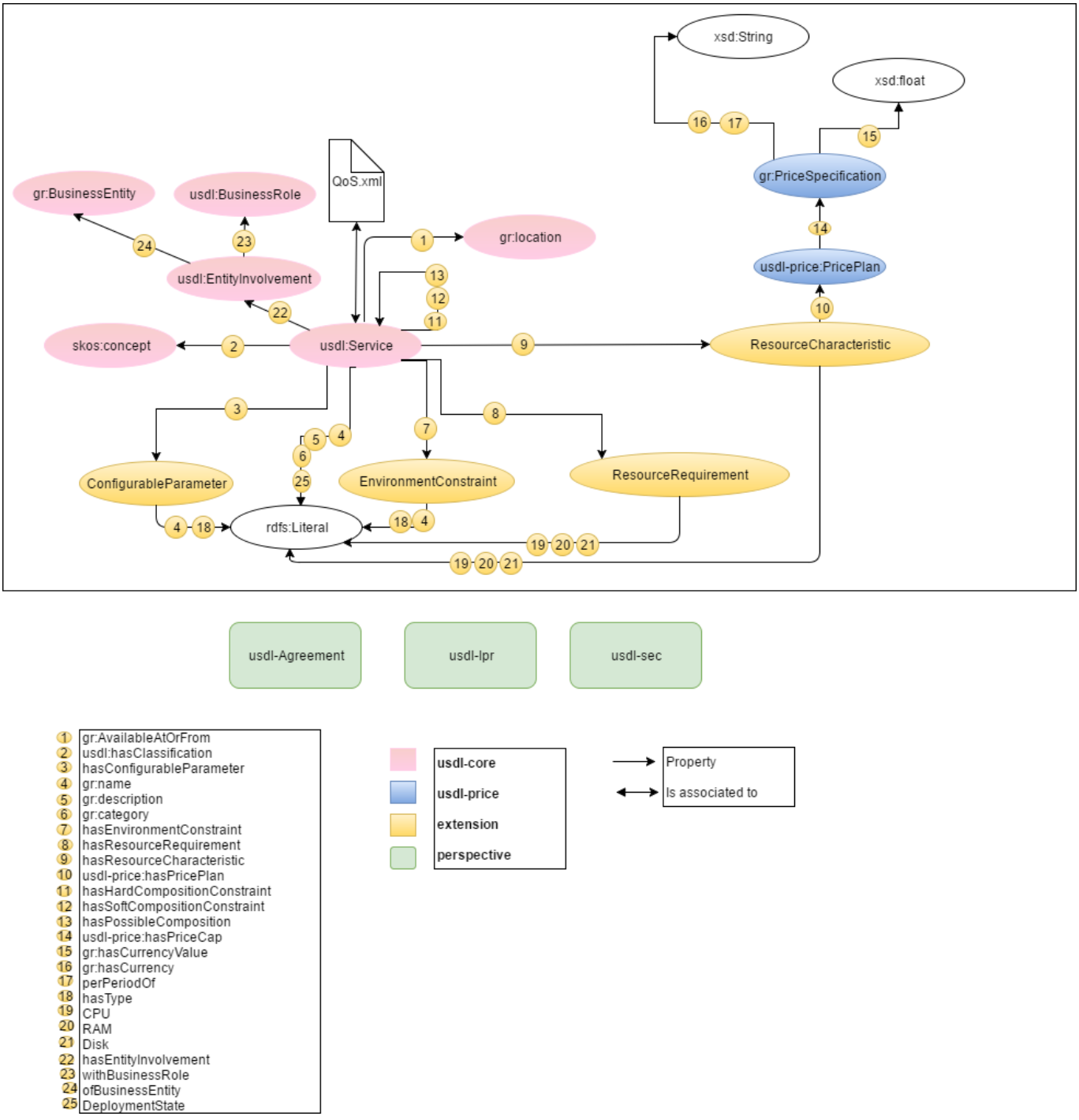

Figure 2. Extended Linked USDL

4.1.2 Deployment Constraints

Deployment constraints cover environment and resources constraints.

Environment constraints

Each business service requires a specific deployment environment. Environment constraints have been described in the work of (Nguyen, Lelli, Papazoglou, & Heuvel, 2012). However, the resource and composition constraints are grouped under the same concept, which is not useful when

automating the composition and the deployment of business services. For automation purposes, we define the environment and resource constraints' concepts separately for each business service.

Definition 3. Environment constraints of a business service (property 7 in Figure 2) represent software that must be installed on the virtual machine hosting the business service described. Each environment constraint is described by a type (e.g., Web server) and name (e.g., Apache). We assume that environment constraints are automatically incorporated into the deployment scripts of a business service.

Resource Constraints

Each business service requires a minimum of resources to ensure its normal functioning.

Definition 4. Resource constraints (property 8 in Figure 2) of a business service represent the characteristics required for the virtual machine hosting this service in terms of CPU (property 19 in Figure 2), memory (Property 20 in Figure 2), and disk (Property 21 in Figure 2).

4.1.3 IaaS technical characteristics

To support the selection of an IaaS that satisfies the deployment constraints of a business service, we define for each IaaS (e.g., Amazon EC2) the technical characteristics of the instances it offers (property 9 of Figure 2). A cost plan is associated with technical characteristics (property 10 of Figure 2) describing the cost of virtual machine instance for a given supplier.

4.1.4 Configurable parameters

To automate the configuration of a business application we define the configurable parameters of each service.

Definition 5. Configurable parameters (property 3 in Figure 2) of a business service represent parameters that can be customized for the use of the service, such as application name, logo etc. Each configurable parameter is described by its name ("gr: name", property 4 in Figure 2) and by the type of the HTML component (property 18 in Figure 2) to be inserted into the configuration interface (e.g., a text box for the name of an application and a browse button for the logo of the application).

4.1.5 Business service category

For each business service, we define its category using "gr:category" property (property 6 in Figure 2). A category is the family of services to which a service belongs. For example, MediaWiki service belongs to "Wiki engine" family. This notion will be used to evaluate the quality of services with respect to other similar services.

4.1.6 Quality of service settings

We choose in this work to entrust the task of describing QoS parameters to a third-party service. Many third-party services for service evaluation and comparison are available on the Web such as Cloud Armor (Cloud Armor) and Cloudorado (Cloudorado). The former provides a dataset of QoS ranks (e.g., availability, response time, and ease of use) assigned by users to the cloud services used. The latter provides a comparison of cloud providers in terms of SLA level, price, and functionality. To illustrate the description of a service's quality we consider four parameters namely respect of data confidentiality, preservation against data loss, availability of the service, and response time of the service. Listing 1 illustrates an example of a returned XML file, describing the QoS parameters of a given service.

```
1. <?xml version ="1.0" encoding="UTF-8" standalone="no"?>
2. <QoS>
3. <Availability >7</Availability>
```

```
<Response_Time>173.82358</Response_Time>
<Data_Privacy>8</Data_Privacy>
<Data_Loss>8.902313</Data_Loss>
</QoS>
```

Listing 1. XML file describing the QoS parameters of a service

4.1.7 Deployment state

The deployment state is modelled using property 25 in Figure 2. It concerns business services. Two deployment states are considered: deployed and deployable.

4.2 Extending Linked USDL for user's requirements description

The requirements description implies the gathering, transformation, and treatment of the user's needs for the desired business application. In our work, we allow the user to express her requirements via a Web form since we are interested in the type of considered requirements instead of how the requirements are collected. We have identified two major goals for requirements identification:

- **Goal 1**: Identify the minimum requirements for selecting and composing business services that satisfy the user's needs.
- **Goal 2**: Consider non-technical requirements that may be important to the user, and which may allow to select one service composition plan over another. Existing Linked USDL specification considers non-technical requirements, such as business ones, however it lacks the service user needs, like for example the required QoS characteristics.

To achieve these goals, we define the RequIrements VocAbuLary (RIVAL) as a new module of Linked USDL to formalize the functional and non-functional user's requirements. RIVAL reuses existing concepts from the Good Relations vocabulary (GoodRelations: The professional Web vocabulary for e-commerce, 2008), RDFS (RDF Schema 1.1, 2014), and XSD (XML Schema, 2004). It also introduces new concepts allowing to select the services meeting the user's functional requirements and their composition possibilities. Figure 3 illustrates RIVAL classes that describe the vocabulary's concepts, and properties that describe relationships between classes. To reduce the technical knowledge required for the provisioning of cloud applications, no technical requirement is asked to the user.

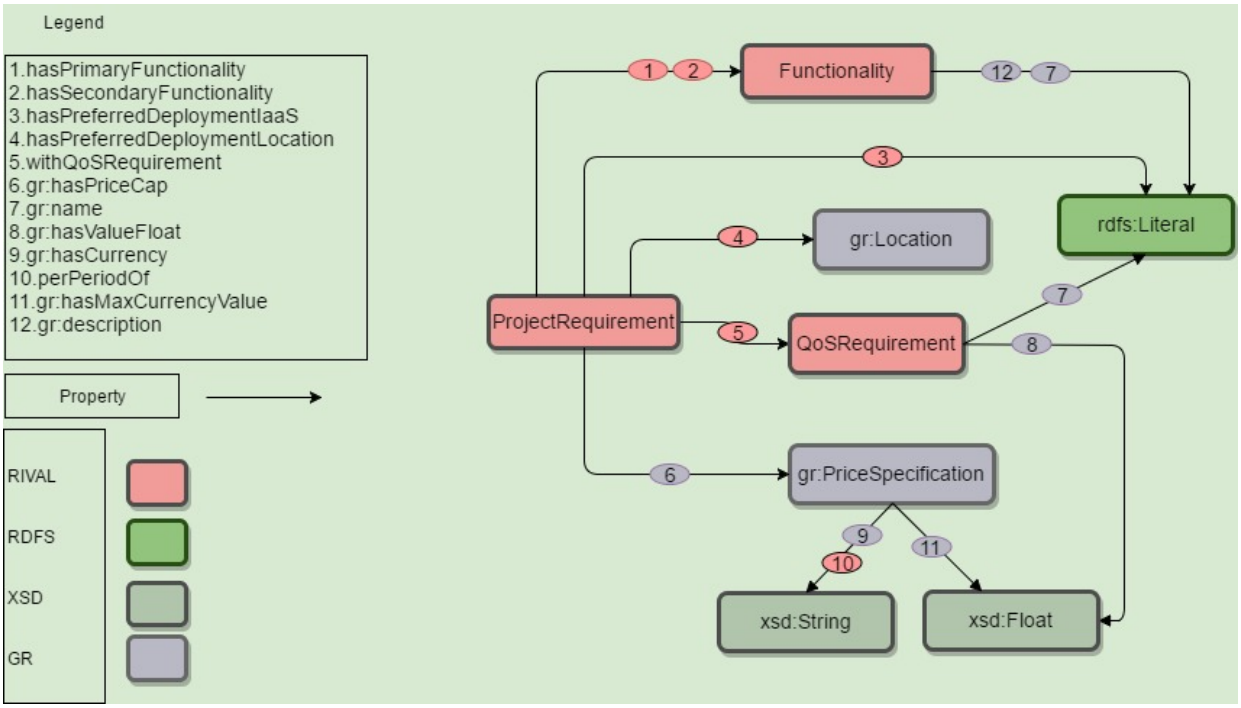

Figure 3. RIVAL's module overview

4.2.1 Functional requirements

Functional requirements meet the primary objective of the required service. They represent the global functionalities that the business application must accomplish. These are described either by keywords describing the desired functionality or objective, or by the name of the business service that satisfies them. In this work, the description of functional requirements excludes any technical details such as deployment or composition constraints. We introduce a distinction between primary and secondary functionalities (Definitions 6 and 7) which guide respectively the selection of primary services and their composition possibilities (secondary services).

Definition 6. A primary functionality (property 1 in Figure 3) describes the overall functionality for a business application desired by the user. A primary functionality is independent from each other, and can be linked to secondary functionalities.

Definition 7. A secondary functionality (property 2 in Figure 3) is related to a primary functionality, and enhances the functionality of the latter, but it is not essential in the business application.

A functionality of a desired application is provided by a business service and is considered as primary, e.g., project management. Any additional functionality to this project management is considered as secondary, e.g., requiring a version management service with the project management one. Only one primary functionality is allowed in user's requirements. Several secondary functionalities can be associated with it.

4.2.2 Non-functional requirements

Non-functional requirements satisfy the second objective and cover the requirements in terms of QoS as well as the user preferences with regard to the desired business application. It is difficult for a user to estimate acceptable tolerance thresholds for QoS parameters such as data availability or integrity. In fact, usually users always aim for maximum quality. For these reasons, we use weights (Definition 8) that the user assigns to the QoS parameters. The quality of service requirements for the desired application are therefore described in RIVAL by their names (property 7 in Figure 3) and their weights that are assigned by the user (property 8 in Figure 3).

Definition 8. The weight assigned to each quality of service parameter describes the priority that the user assigns to it. We decided to allow the user to distribute 10 points between the considered QoS parameters, so that the sum of all the assigned values is equal to 10 (the values can be integers or decimals). The choice of the sum equals to 10 is due to the simplicity, in our sense, to distribute 10 points rather than a percentage. These weights will be used to evaluate the quality of the discovered services.

The description of QoS weights is optional. In the case where the user does not determine her priorities for QoS parameters, the same weight will automatically be assigned by default to each of considered parameters.

User preferences are related to deployment and payment details, including:

- The deployment location (property 4 in Figure 3): the user can choose the continent where her application will be deployed. This requirement is optional, but may be important for the user when she values the sensitivity of her data or the privacy laws in different continents.
- The name of the IaaS provider which will host the desired application (property 3 in Figure 3). This preference is based on a previous experience of using an IaaS for hosting the application. This requirement is optional.
- Payment details are described by the "PriceSpecification" class of the "Good Relations" vocabulary (GoodRelations, 2008), which is associated with a currency (property 9 in

Figure 3), a maximum cost (property 11 in Figure 3), and a billing period (property 10 in Figure 3).

5 VALIDATION AND EVALUATION

The extended Linked USDL has been used with MADONA to describe respectively the marketplace's services and a user's requirements (Benfenatki et al., 2016). MADONA has been implemented and a video of the system is available at liris.cnrs.fr/hind.benfenatki/demo.mp4.

Let us consider the following scenario to illustrate the use and the benefit of the extended Linked USDL while provisioning cloud-based service-oriented business applications: A manager in a medical clinic (herein the user) wants to provision an application which is capable to manage patient records and medical procedures billing. In this scenario, a patient records management functionality represents the primary functionality and procedures billing functionality represents a secondary one. From these functional requirements, several composition plans are generated following the composition plans generation algorithm described in Listing 4 in (Benfenatki et al., 2016). Figure 4 illustrates the generated composition plans. Each composition plan bounds a set of relations. Each relation composes a business service with its composition constraints and/or composition possibilities. In fact, the first relation of each composition plan composes a service meeting the user's primary functionality (herein after called a primary service), the services representing the composition constraints of the primary service, and the services meeting the user's secondary functionalities and representing a composition possibility of the primary service. The other relations compose the composition plan's services with their composition constraints. The generation of composition plans is done automatically and dynamically since composition constraints and possibilities are known from service's description using extended Linked USDL.

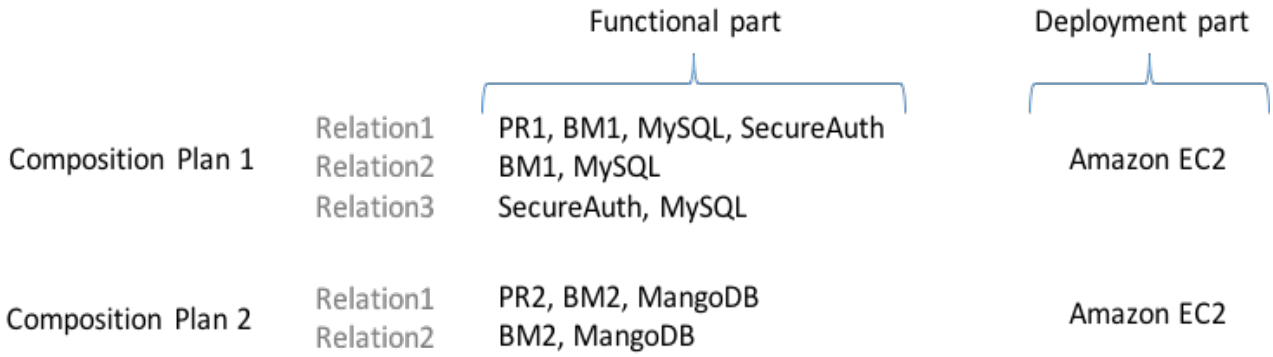

Figure 4. Generated composition plans

The first composition plan in Figure 4 is composed of three relations. The first one composes *PR1*, a service meeting the patient records management functionality, *BM1*, a service meeting billing management functionality, and a *MySQL* database and *SecureAuth* that represent the composition constraints of the primary service, and allow respectively to store patient information and to guarantee a secure authentication to the generated application. The second and third relations compose *BM1* and *SecureAuth* with their composition constraints.

The generated composition plans are completed with an IaaS meeting the user's deployment preferences and QoS requirements. The cost of each composition plan is evaluated for the selected IaaS and the ones exceeding the user's payment preferences are excluded. The remaining composition plans are evaluated according to the user's QoS requirements and services' QoS history. The composition plan with the highest QoS is selected for deployment. For the selected composition plan, several Web forms are displayed to the user so that she can personalize the generated

application. Configuration and deployment scripts are automatically generated and executed. More details on MADONA's phases are reported in (Benfenatki et al., 2016).

We evaluate in Figure 5 the benefit of the extended Linked USDL on the provisioning of the running scenario's generated application (corresponding to the first composition plan with two configurations). On the one hand considering composition constraints while describing a marketplace's services allows to know the minimal composition permitting the normal functioning of a service. Moreover, it allows to automate the composition process, i.e., the generation of composition plans, thus reducing the necessary technical knowledge required from the user for provisioning service-oriented cloud applications. On the other hand, considering configurable parameters and deployment constraints for each business service allows to automate the configuration and the deployment respectively.

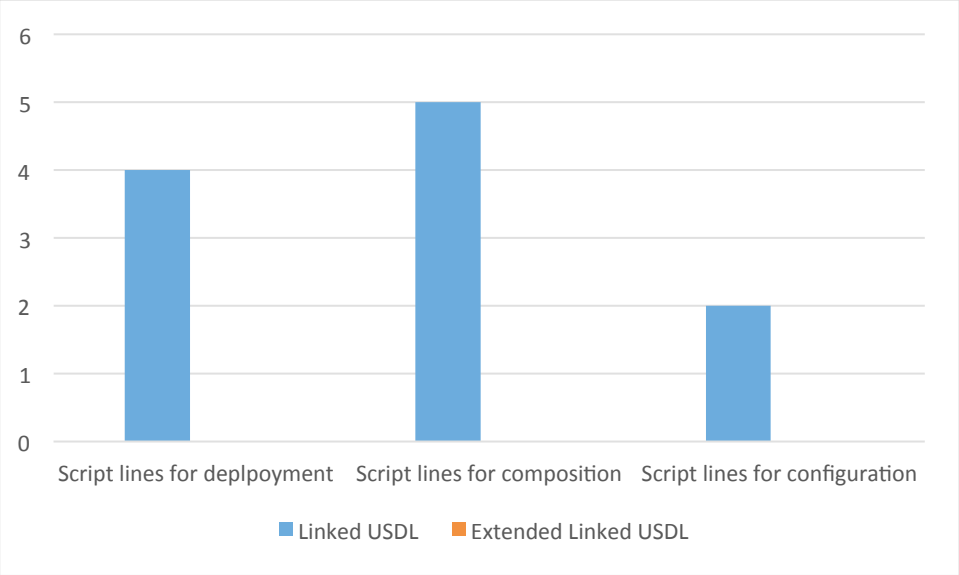

Figure 5. Extended Linked USDL evaluation

The number of invoked services is equal to the sum of the number of desired functionalities (primary and secondary) and the number of composition constraints associated to those functionalities. As shown in Figure 6, the number of services (or of functionalities) the user has to know remains fixed while the number of the associated composition constraints grows. In fact, composition constraints are taken into account automatically from the service's description.

Furthermore, by describing composition possibilities, we reduce the number of generated composition plans. In fact, the generated composition plans compose only the services that can be composed.

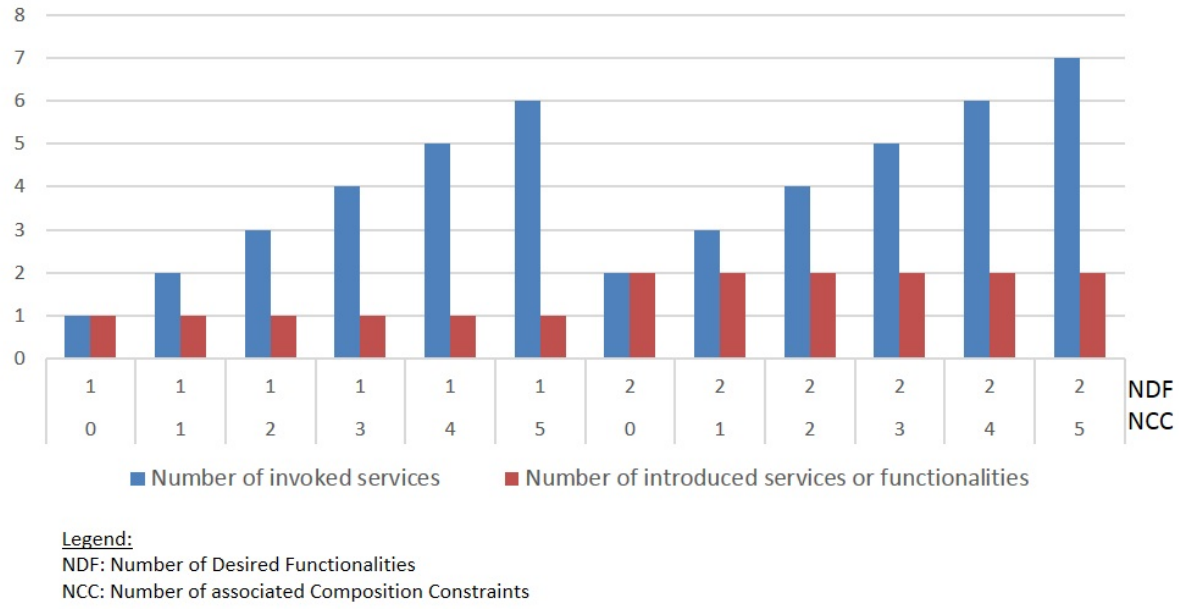

Figure 6. Number of introduced services according to the number of desired functionalities and their associated composition constraints

CONCLUSION

We have defined an extension of Linked USDL for the description of the marketplace's services (business and infrastructure) and for the service requirements of the user. New concepts have been added to describe the relationships that a business service must and can have with other peers in order to know the composability of a service. We also described for each business service its deployment constraints and configurable parameters, in order to automate the deployment and the configuration of a given business service. We have described the technical characteristics of an IaaS service in order to allow the selection of resources responding to the deployment constraints of each business service. We entrust the description of the quality of the marketplace's services to third-party services in order to have an objective representation of the quality of the service.

We also have defined the RIVAL module to formalize the user's requirements, which are described through a Web form. RIVAL defines the minimal requirements, functional and non-functional, allowing an effective selection and composition of business services, and introduces the notion of primary and secondary functionalities. Non-functional requirements include user deployment and cost preferences, and QoS requirements. The requirements taken into account in RIVAL are expressed at a high level of abstraction of technical details. For example, QoS requirements are expressed in terms of weights symbolizing the importance the user assigns to each quality parameter instead of it being expressed in precise values.

We present, as well, the results of experiments demonstrating the use of our extension of Linked USDL with MADONA, a method for automated provisioning of cloud-based service-oriented business applications. We can conclude that Linked USDL extension allows to generate automatically and dynamically composition plans meeting user's functional requirements and meeting services composition constraints and possibilities. In fact, each service of a composition plan is automatically composed with the services representing its composition constraints as the latter are known from service's description. Furthermore, only composable services are composed and this is done automatically. In fact, composition possibilities of marketplace's services are known from their descriptions. Thus, describing composition constraints and possibilities of each business service

allows (i) to automate the composition process, (ii) to consider the minimal composition allowing the good functioning of each service, and (iii) to compose only composable services.

Considering resource constraints while describing a business service allows to automate its deployment on sufficient resources allowing its good functioning. Considering configurable parameters of each marketplace's business service allows to automate the configuration process. Hence, all the application provisioning process is automated.

As part of our ongoing work, we plan to consider a cost model which considers more relevant parameters when estimating the use of resources for deploying business services, for instance, the type of storage and bandwidth associated with the virtual machines deployed. We also plan to consider the cost of business services of the generated application.

REFERENCES

Afify, Y. M., Moawad, I. F., Badr, N. L., & Tolba, M. (2014). Cloud services discovery and selection: Survey and new semantic-based system. *Bio-inspiring Cyber Security and Cloud Services: Trends and Innovations* , 449–477.

Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., et al. (2007). *Web service-business process execution language (ws-bpel)*. From http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

Becha, H., & Amyot, D. (2014). Consumer-centric non-functional properties of soa-based services. *Proceedings of the 6th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems, PESOS 2014* , 18–27.

Benfenatki, H., Ferreira Da Silva, C., Kemp, G., Benharkat, A.-N., Ghodous, P., & Maamar, Z. (2016). MADONA - a method for automated provisioning of cloud-based component-oriented business applications. *Service Oriented Computing and Applications , 11* (1).

Bhasin, H. (2016). *Classification of services*. From http://www.marketing91.com/classification-of-services/

Binz, T., Breitenbücher, U., Kopp, O., & Leymann, F. (2014). Tosca: Portable automated deployment and management of cloud applications. (Springer, Ed.) *Advanced Web Services* , 527–549.

BPMN, Business Process Model and Notation. (2011). From http://www.bpmn.org/

Cardoso, J. (2013). Modeling service relationships for service networks. *Proceedings of the 4th International Conference on Exploring Services Science, IESS2013* , 114–128.

Cardoso, J., & Pedrinaci, C. (2015). Evolution and overview of linked usdl. *Proceedings of the 6th International Conference on Exploring Services Science, IESS2015* , 50–64.

Cardoso, J., & Pedrinaci, C. (2015). Evolution and Overview of Linked USDL. *International Conference on Exploring Services Science* , 50-64.

Cardoso, J., Barros, A., May, N., & Kylau, U. (2010). Towards a unified service description language for the internet of services: Requirements and first developments. *IEEE International Conference on Services Computing (SCC)* , 602-609.

Cardoso, J., Binz, T., Breitenbücher, U., Kopp, O., & Leymann, F. (2013). Cloud computing automation: integrating usdl and tosca. *Proceedings of the 25th International Conference on Advanced Information Systems Engineering, CAISE2013* , 1–16.

Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. (2001). Web services description language (WSDL) 1.1.

Cloud Armor. (n.d.). Retrieved in 2017 from Cloud Armor Project Website: cs.adelaide.edu.au/~cloudarmor/

Cloud service measurement index consortium. (2016). Retrieved in 2016 from http://csmic.org/

Cloudorado. (n.d.). Retrieved 2017 from Cloud computing price comparison | Cloudorado - Find best cloud server from top cloud computing companies: https://www.cloudorado.com

Dbpedia. (2016). From http://wiki.dbpedia.org/

D'Ambrogio, A. (2006). A model-driven wsdl extension for describing the qos of web services. (IEEE, Ed.) *International Conference on Web Services, ICWS2006* , 789–796.

GoodRelations. (2008). From GoodRelations: The professional Web Vocabulary for E-Commerce: http://www.heppnetz.de/projects/goodrelations/

GoodRelations: The professional Web vocabulary for e-commerce. (2008). From http://www.heppnetz.de/projects/goodrelations/

International classification of goods and services for the purposes of the registration of marks. (2001). From http://www.wipo.int/export/sites/www/classifications/nice/en/pdf/8_list_class_order.pdf

Kan, J., & Sim, K. M. (2011). Cloudle: an ontology-enhanced cloud service search engine. *Web Information Systems Engineering–WISE 2010 Workshops* , 416–427.

Lagares Lemos, A., Daniel, F., & Benatallah, B. (2015). Web service composition: A survey of techniques and tools. (ACM, Ed.) *ACM Computing Surveys* .

Linked data. (2015). From Linked data - connect distributed data across the web: http://www.linkeddata.org/

Linked USDL. (2015). Retrieved 2015 from Linked usdl: http://linked-usdl.org/

Linked USDL modules. (2015). From http://github.com/linked-usdl/

Liu, D., & Zic, J. (2011). Cloud#: A specification language for modeling cloud. *Proceedings of the International Conference on Cloud Computing, CLOUD2011* , 533–540.

Lüftenegger, E. (2014). Service-dominant business design. *Eindhoven University of Technology* .

Maamar, Z., Wives, L. K., Badr, Y., Elnaffar, S., Boukadi, K., & Faci, N. (2011). Linkedws: A novel web services discovery model based on the metaphor of "social networks". *Simulation Modelling Practice and Theory* , 121–132.

Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., et al. (2004). Bringing semantics to web services: The owl-s approach. *First International Workshop on Semantic Web Services and Web Process Composition* , 26–42.

Mitra, G., Zhou, X., Bouguettaya, A., & Liu, X. (2013). A request oriented model for web services. *Proceedings of the 1st Australasian Web Conference* , 13–20.

Nguyen, D. K., Lelli, F., Papazoglou, M. P., & Heuvel, W.-J. v. (2012). Issue in automatic combination of cloud services. (IEEE, Ed.) *10th International Symposium on Parallel and Distributed Processing with Applications, ISPA2012* , 487–493.

RDF Primer. (2004). From https://www.w3.org/TR/2004/REC-rdf-primer-20040210/

RDF Schema 1.1. (2014). From https://www.w3.org/TR/rdf-schema/

Turtle recommandation. (2014). *Turtle*. From https://www.w3.org/TR/turtle/

Semantic annotations for WSDL and XML schema. (2007). Retrieved 2015 from http://www.w3.org/TR/sawsdl/

Services. (2016). *What is human services?* Retrieved 2016 from http://www.nationalhumanservices.org/what-is-human-services

SPARQL Protocol for RDF. (2008). From https://www.w3.org/TR/rdf-sparql-protocol/

RDF. (2014). *Resource Description Framework (RDF)*. From https://www.w3.org/RDF/

Juju Charms. (2016). From https://jujucharms.com/store

Sun, W., Zhang, K., Chen, S.-K., Zhang, X., & Liang, H. (2007). Software as a service: An integration perspective. (Springer, Ed.) *Proceedings of the International Conference on Service-Oriented Computing, ICSOC2007* , 558–569.

Taekgyeong, H., & Sim, K. M. (2010). An ontology-enhanced cloud service discovery system. *Proceedings of the International MultiConference of Engineers and Computer Scientists* , 17–19.

Tahamtan, A., Beheshti, S. A., Anjomshoaa, A., & Tjoa, A. M. (2012). A cloud repository and discovery framework based on a unified business and cloud service ontology. *Proceedings of the 8th World Congress on Services, SERVICES2012* , 203–210.

Taher, Y., Nguyen, D. K., Lelli, F., Heuvel, W.-J. v., & Papazoglou, M. (2012). On engineering cloud applications-stateof the art, shortcomings analysis, and approach. *Scalable Computing: Practice and Experience* , 215–232.

Tsai, W.-T., Sun, X., & Balasooriya, J. (2010). Service-oriented cloud computing architecture. *7th International Conference on Information Technology: New Generations, ITNG2010 , IEEE*, 684–689.

USDL. (2011). From Unified service description language: https://www.w3.org/2005/Incubator/usdl/

Web Service Modeling Ontology (WSMO). (2008). From http://www.wsmo.org/

WS-CDL. (2005). From Web service-choreography description language (ws-cdl): https://www.w3.org/TR/ws-cdl-10/

WSDL. (n.d.). Retrieved 2016 from Web service description language (wsdl): http://www.w3schools.com/xml/ xml_wsdl.asp

Xiang, J., Liu, L., Qiao, W., & Yang, J. (2007). Srem: A service requirements elicitation mechanism based on ontology. (IEEE, Ed.) *Proceedings of the 31st Annual International Conference on Computer Software and Applications, COMPSAC2007* , 196–203.

XML Schema. (2004). From https://www.w3.org/XML/Schema

Yuan, X., & Zhang, X. (2015). An ontology-based requirement modeling for interactive software customization. (IEEE, Ed.) *International Model-Driven Requirements Engineering Workshop, MoDRE2015* , 1–10.

Zhang, M., Ranjan, R., Haller, A., Georgakopoulos, D., Menzel, M., & Nepal, S. (2012). An ontology based system for cloud infrastructure services discovery. *Proceedings of the 8th International Conference on Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom2012* , 524–530.