Cloud Computing Automation: Integrating USDL and TOSCA

Jorge Cardoso[1,2], Tobias Binz[3], Uwe Breitenbücher[3], Oliver Kopp[3], and Frank Leymann[3]

[1] Karlsruhe Service Research Institute (KSRI)
Karlsruhe Institute of Technology, Karlsruhe, Germany
[2] CISUC, Department of Informatics Engineering
University of Coimbra, Coimbra, Portugal
jcardoso@dei.uc.pt
[3] Institute of Architecture of Application Systems
University of Stuttgart, Stuttgart, Germany
{lastname}@iaas.uni-stuttgart.de

Abstract. Standardization efforts to simplify the management of cloud applications are being conducted in isolation. The objective of this paper is to investigate to which extend two promising specifications, USDL and TOSCA, can be integrated to automate the lifecycle of cloud applications. In our approach, we selected a commercial SaaS CRM platform, modeled it using the service description language USDL, modeled its cloud deployment using TOSCA, and constructed a prototypical platform to integrate service selection with deployment. Our evaluation indicates that a high level of integration is possible. We were able to fully automatize the remote deployment of a cloud service after it was selected by a customer in a marketplace. Architectural decisions emerged during the construction of the platform and were related to global service identification and access, multi-layer routing, and dynamic binding.

Keywords: USDL, TOSCA, cloud service lifecycle, service description, service management.

1 Introduction

Standardization efforts are paving the way which leads to the mainstream adoption of SaaS (Software-as-a-Service) and cloud computing environments [1]. Currently, different players (e.g., OMG, W3C, Eurocloud, NIST) are undertaking several initiatives[1] (e.g., USDL, TOSCA, CCRA, OCCI) to provide useful and usable standards for cloud computing. In 2009, it was argued that no standard existed [2]. This has changed. For example, The Open Group is working on the Cloud Computing Reference Architecture (CCRA) and EuroCloud is devising guidelines on law, data privacy, and compliance.

[1] http://cloud-standards.org

C. Salinesi, M.C. Norrie, and O. Pastor (Eds.): CAiSE 2013, LNCS 7908, pp. 1–16, 2013.

Nonetheless, these initiatives have two limitations. On the one hand, efforts are being conducted in isolation and it is not clear to which extend they can be integrated and, on the other hand, there is a lack of certainty as to which standards provide adequate levels of interoperability. For cloud providers (e.g., `HostEurope.com` and `JiffyBox.de`), advances in interoperability can simplify the countless activities involved during the life cycle of applications.

The objective of this paper is to study to which extend current cloud specifications and standards are interoperable. In particular, we investigate how USDL (Unified Service Description Language) [3,4] and TOSCA (Topology and Orchestration Specification for Cloud Applications) [5] can be integrated to link the description and the management of cloud services[2], respectively. USDL is being explored by several research projects to enhance the description of service offerings to facilitate service discovery and selection [6]. TOSCA helps providers to automate the deployment and management of services.

Our research design uses the SaaS application SugarCRM[3], an open-source, web-based customer relationship management (CRM) platform, as a representative use case for evaluating the interoperability level of USDL and TOSCA. Therefore, the various SugarCRM service offerings were modeled with their pricing models, software options, and legal statements in USDL. The SugarCRM deployment, which included virtual machines, databases, and web servers, as well as its management, was modeled with TOSCA. Based on these activities, the development of a loosely coupled platform as a mean to achieve interoperability between the two specifications was conducted, building the core part of the proposed approach. The development of the platform, called SIOPP (ServIce Offering and Provisioning Platform)[4], involved taking architectural decisions to enable the global and unique identification of services described with USDL, the remote access and querying of USDL service descriptions, the intelligent routing of service requests to providers, and the dynamic binding of TOSCA deployment descriptors to service descriptions.

The evaluation of the platform indicated that a high degree of interoperability was achieved. It became possible to select a cloud service from a marketplace, route the request to a provider which had previously announced to offer the service, and deploy the cloud service using plans which accounted for the characteristics of the service. After setup and configuration, all these steps were conducted automatically without requiring human intervention. Future work requires the replication of our research using other emerging specifications (e.g., CloudAudit for auditing and BSI-ESCC for security) to support the full life cycle of cloud applications from *cradle to grave*.

This paper is structured as follows: In Section 2, we illustrate a motivating scenario explaining the need to integrate cloud specifications. Section 3 explains how the SaaS SugarCRM from our scenario was described using USDL and how its deployment was specified using TOSCA. The requirements for a platform to make USDL and TOSCA interoperable, as well as the main architectural

[2] We will use the terms service and cloud application to refer to Software-as-a-Service.
[3] `http://www.sugarcrm.com/`
[4] SIOPP is pronounced 'shop'.

decisions are presented in Section 4. Section 5 evaluates the developed platform. Section 6 provides a literature review. Section 7 discusses our conclusions.

2 Motivating Scenario

Nowadays, the discovery and selection of cloud applications, such as a SaaS SugarCRM system, is still mainly carried out manually by consumers. It is not possible to effectively query services offered by different marketplaces (e.g., AppDirect, Appcelerator, and the Service Delivery Broker from Portugal Telecom), because they are not publicized using computer-understandable formats. Marketplaces need to be searched manually. This is a first limitation we want to address.

After a purchase decision is made, and from the provider side, contracting and billing is negotiated by the sales and procurement divisions, the selected cloud application and its customization is given to an IT provider or department without any formalization of the executables, technical requirements, management best practices, and so on. Operators invest considerable efforts to learn how to setup and manage the application. Customization is done manually and often research or consulting is required to make a cloud solution work in a particular environment. This manual and error-prone process is not suitable to address fast changing markets and dynamic business requirements. Apart from solutions such as Saleforce, Google Apps, or Microsoft Office 365, this is still the way software is provisioned. This is the second limitation we want to address.

To solve these limitations, USDL is aiming to formalize, structure, and simplify the discovery and selection of services, and TOSCA to automate their management. When used in conjunction, they can automate parts of the lifecycle of cloud applications, namely discovery, selection, deployment, and management.

3 Modeling SugarCRM with USDL and TOSCA

In this section we provide a brief introduction to the two specification languages we will integrate. We also use USDL to describe the SaaS SugarCRM application from our scenario and use TOSCA to model its deployment.

3.1 USDL Overview

The Unified Service Description Language was developed in 2008 for describing business, software, or real world services using machine-readable specifications to make them tradable on the Internet [3]. Past efforts were concentrated on developing languages, such as WSDL, CORBA IDL, and RPC IDL, which focused on the description of software interfaces. Nonetheless, the Internet of Services requires services to be traded, placing emphasis on the description of business-related aspects such as pricing, legal aspects, and service level agreements. This was the motivation to create USDL. The initial versions of USDL were ready in 2009 [7,3]. Later, in 2011, based on the experiences gained from the first developments, a W3C Incubator group[5] was created and USDL was extended. The

[5] `http://www.w3.org/2005/Incubator/usdl/`

extensions resulted from the experience gained in several European academic and industrial projects (e.g., SOA4ALL, Reservoir, ServFace, Shape, etc.). In 2012, a new version named Linked USDL based on Linked Data principles [8] and RDF was proposed. This recent version is currently being explored and evaluated in several research projects such as FI-Ware (smart applications), FInest (logistics), and Value4Cloud (value-added cloud services).

Linked USDL is segmented in 5 modules. The `usdl-core` module models general information such as the participants involved during provisioning and service options such as customer support. The cost and pricing plans are modeled with `usdl-price`. The legal terms and conditions under which services may be consumed are modeled with `usdl-legal`. The module `usdl-sla` gathers information on the levels of service provided, e.g., availability, response time, etc. Finally, `usdl-sec` models security features of a service. Due to its benefits, e.g., reusability of existing data models and simplicity in publishing and interlinking services, Linked USDL was used in this research.

3.2 Describing SugarCRM with USDL

The information used to model the SaaS SugarCRM was retrieved from its web site. A service and a vocabulary model were created. The vocabulary contained domain dependent concepts from the field of CRM systems (e.g., taxonomies of common installation options). Since Linked USDL only provides a generic service description language, domain specific knowledge needs to be added to further enrich the description of services. The excerpt from Listing 1.1 illustrates the description of the SugarCRM service (in this paper, examples are written using the Turtle language[6]).

```
1  <#service_SugarCRM> a usdl:Service ;
2        ...
3     dcterms:title "SugarCRM service instance"@en ;
4     usdl:hasProvider :provider_SugarCRM_Inc ;
5     usdl:hasLegalCondition :legal_SugarCRM ;
6     gr:qualitativeProductOrServiceProperty
7       crm:On_premise_or_cloud_deployment ,
8       crm:Scheduled_data_backups ,
9       crm:Social_media_integration ,
10      crm:Mobile_device_accessibility .
11      ...
```

Listing 1.1. SugarCRM service modeled with Linked USDL

The description starts with the identification of the provider (line 4), the legal usage conditions (line 5), and the general properties of the service (e.g., deployment, scheduled backups, integration, and mobile accessibility). Service offerings

[6] Turtle – Terse RDF Triple Language, see `http://www.w3.org/TR/turtle/`

connect services to price plans. Listing 1.2 defines four price plans (lines 4-8): `professional`, `corporate`, `enterprise`, and `ultimate`. The `professional` plan includes common features shared with the other plans such as sales force automation, marketing automation, and support automation (lines 15-20). It costs $30 per month (lines 21-25), the contract is annual and the billing is made every month (not shown in this excerpt).

```
1 :offering_SugarCRM a usdl:ServiceOffering ;
2     ...
3     usdl:includes <#service_SugarCRM> ;
4    usdl:hasPricePlan
5     :pricing_SugarCRM_Professional ,
6     :pricing_SugarCRM_Corporate ,
7     :pricing_SugarCRM_Enterprise ,
8     :pricing_SugarCRM_Ultimate ;
9    usdl:hasServiceLevelProfile :slp_SugarCRM .
10 ...
11 :priceComponent_SugarCRM_Professional_General a price:PriceComponent ;
12  dcterms:title "General price"@en ;
13  dcterms:description "Fee for general usage of the instance."@en ;
14  price:isLinkedTo
15    crm:Sales_Force_Automation ,
16    crm:Support_Automation ,
17    crm:Integration_via_web_services_API ,
18    crm:Customizable_Reporting ,
19    ...
20    crm:MySQL_and_MS_SQL_server_database_support ;
21 price:hasPrice
22   [ a gr:UnitPriceSpecification ;
23    gr:hasCurrency "USD" ;
24    gr:hasCurrencyValue "30" ;
25    gr:hasUnitOfMeasurement "MON" ] .
```

Listing 1.2. Pricing plans for SugarCRM services

In this example, Linked USDL uses existing vocabularies such as Dublin Core (shown in the model with `:dcterms`), GoodRelations (`:gr`), and the domain vocabulary constructed for CRM systems (`:crm`).

3.3 TOSCA Overview

The Topology and Orchestration Specification for Cloud Applications [5] was standardized to enable automated deployment and management of applications while being portable between different cloud management environments [9]. The management and operation of cloud applications are major concerns in enterprise IT. For example, the pay-as-you-go model requires fast provisioning and management of application instances. Since these applications typically consist

of numerous heterogenous distributed components, the management of the components itself, the relationships among each other, and the whole application is difficult and expensive in terms of time and money - especially when manual work is required, e.g., deploying and executing scripts in a special order by hand which is error prone. Thus, there is the need to automate management to decrease the effort and reduce the error rate. In addition, to avoid vendor lock-in, which is a major concern of customers when talking about outsourcing and cloud computing, there is a need to create portable applications which can be moved between different cloud providers. The TOSCA specification is currently standardized by an OASIS Technical Commitee[7] which already published a number of community specification drafts. TOSCA is an XML-based exchange format. The application's architecture, the components it consists of, and the relationships among them are modeled formally in a typed topology graph. Each node and relationship defines the management operations it offers. These operations are exposed as web services and are used to manage the individual components and relationships on a fine-granular technical level. The overall management functionalities such as deploying, scaling, backuping, and terminating the whole application are modeled on a higher level of abstraction by using management plans. Plans are implemented as workflows, e.g., in BPMN or BPEL, to benefit from compensation, recovery, and transaction concepts [9].

TOSCA Service Archives package cloud applications with all the required software artifacts such as installables or applications files as well as their management plans in a portable fashion. These archives can be installed in TOSCA Runtime Environments which provide all functionalities to manage the archive and execute management plans. This enables cloud providers to offer third party services because management details, e.g., how to scale the application or how security is achieved, are hidden and the archives can be treated and operated as a self-contained black box. As the specification does not define a visual notation, in this paper we use Vino4TOSCA [10] as a visual notation for TOSCA.

3.4 Modeling SugarCRM with TOSCA

In this section we show how the SugarCRM deployment was specified with TOSCA, discuss different deployment options, and list possible variabilities. Figure 1 shows one possible topology of a SugarCRM deployment.

The core components of the application are the `SugarCrmApp`, which is a PHP application, and the `SugarCrmDb` representing the database used by SugarCRM, indicated by the `MySqlDbConnection`. The PHP application requires an Apache web server including a PHP runtime, which is provided by the installed PHP module. To provide the database, a MySQL relational database management system (`MySQLRDBMS`) is used. Currently, SugarCRM also supports Microsoft SQL, Oracle 11g, and IBM DB2 which could be used in other deployment options. Apache and MySQL themselves must be installed on an operating system which is in turn provided as a virtual machine image. All nodes have properties, not

[7] `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca`

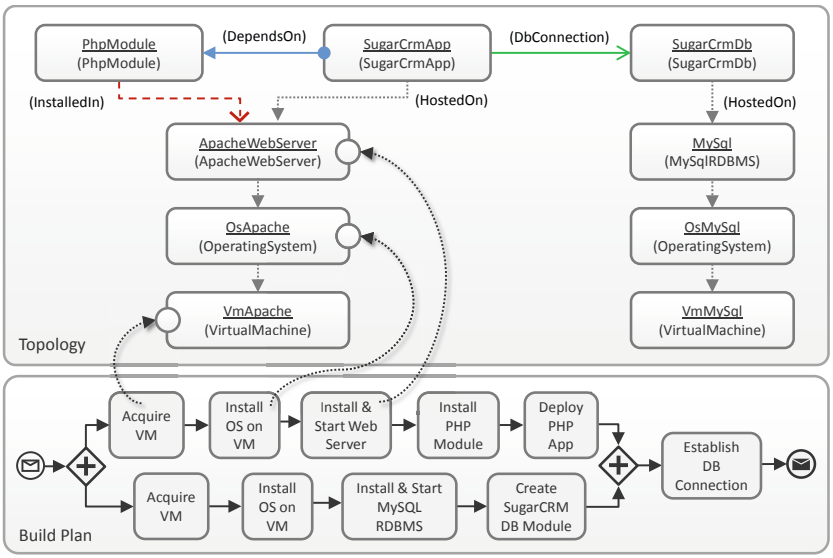

Fig. 1. TOSCA Service Archive containing topology (top) and build plan (bottom) for SugarCRM `ultimate`

explicitly depicted in the figure, holding state and management information of the respective nodes. These properties are used to store information about the application: static information such as the hardware specification of a virtual machine, as well as runtime information such as IP-addresses. This information is used and stored by plans during deployment, management, and termination of the application. The fine grained decomposition into components is needed to understand the interdependencies and variabilities exposed via Linked USDL.

One option to support different SugarCRM offerings is to use separate TOSCA topologies with different quality of service (QoS) captured by USDL service offerings. The `ultimate` deployment depicted in Figure 1 (upper box), for example, hosts the web server and database on different virtual machines, whereas an `enterprise` deployment can use the same virtual machine for both.

On the other hand, there are variations which do not change the structure of the topology. For example, aspects like support options and variations impacting how the application is technically provided. For the latter, possible configurations offered by the VM node are the cloud provider, e.g., Amazon or Rackspace, the physical location, e.g., US or Europe, as well as CPU power, memory size, and hard disk capacity. Beside nodes, it is also possible to configure relations. For example, to tackle security issues, the database connection may be encrypted.

Management plans read and write properties which hold runtime information of nodes and relationships. TOSCA designates one plan as build plan, which deploys and initializes the service. Figure 1 (lower box) shows a simplified example of a build plan which sets up the `ultimate` version of SugarCRM based on two virtual machines. A real executable plan needs additional activities and structural components for data handling, compensation, and recovery mechanisms. The

shown plan sets up the two infrastructure stacks in parallel starting from the bottom by calling management operations provided by the nodes (Figure 1 depicts only three of these calls for simplicity reasons). After stacks are instantiated, the database connection is established and the application is available.

4 USDL and TOSCA Interoperability

Our study on USDL and TOSCA interoperability is timely and relevant because despite standardization efforts clouds may develop in a way that lacks interoperability, portability, and reversibility, all crucial for the avoidance of lock-in. Our approach connects in a best of breed manner two promising standardization efforts, focusing on different aspects, and proposes an end to end solution for cloud services including modeling, discovery, selection, deployment, and management.

A simple solution to integrate USDL and TOSCA consists in establishing a static link between service descriptions and their corresponding archives. Nonetheless, since this approach is strongly coupled it would not be able to handle the dynamics of a global service distribution network. For example, what would happen if the TOSCA descriptor associated with a USDL service description would no longer be valid? What if the deployment provider has ceased its operations and transferred its obligations to, presumably, some other provider which will still handle the original function? How should the request be handled?

4.1 Architectural Decisions

Engineering a platform to integrate service descriptions with service deployments is a major undertaking [6]. We require an architecture that enables a simple transmission of service requests and deployment information between customers and providers via marketplaces; which handles adding or removing marketplaces and providers in a loosely coupled manner; which uses a standard data representation and querying format to ease information exchange and enable interoperability; and which can rely on existing applications, tools and technologies. When examining theoretical and technological advancements to serve as a fundamental building block it becomes clear that the World-Wide Web combined with semantic web technologies is a potential candidate. It is distributed, scalable, reliable, extensible, simple, and equitable [11]. Therefore, the integration platform developed was constructed based on three main underlying principles:

1. Global service identification and service description access,
2. Intelligent routing of service requests, and
3. Dynamic binding of deployment descriptors.

The description of cloud services using Linked USDL provides a global service identification mechanism by using HTTP URIs. It also provides a global, standard, and uniform data access [12] to service descriptions by using HTTP URLs and RDF. In contrast to other approaches, e.g., APIs provided as REST or WS-* endpoints [13], an uniform data access enables a simpler interoperability

and integration of the marketplace, containing service descriptions, and service providers' platforms responsible for SaaS deployment and management.

The routing of service requests from marketplaces to providers is achieved using an intelligent content-based routing [14]. The analysis of Linked USDL descriptions is implemented through SPARQL and can also make use of RDF-based reasoning engines (e.g., Jena, Pellet, FaCT). Their use for content-based routing enables a more flexible routing mechanism compared with web APIs, because full remote access and querying of the service descriptions is possible. Furthermore, the use of a routing mechanism decouples space and time between marketplaces and providers.

Cloud providers use a publish-subscribe pattern [15] to establish a dynamic binding of deployment descriptors with Linked USDL service offerings. This enables cloud providers to quickly adapt to peak demand by scaling the number of servers which handle deployment requests using TOSCA Runtime Environments.

These architectural considerations are evaluated in Section 5 with the implementation of the ServIce Offering and Provisioning Platform (SIOPP).

4.2 Global Service Identification and Description Access

Cloud applications, such as the SugarCRM of our scenario, can be advertised in marketplaces [3] (e.g., SAP Service marketplace, Salesforce.com, and AppDirect.com), or in any other system answering to HTTP URIs requests (e.g., the provider's web sites), which enables consumers to browse through various offerings. A marketplace, or information system, is said to be USDL-compliant if all service offerings are modeled with Linked USDL, and are externally visible and accessible via HTTP URIs. Since Linked USDL relies on linked data principles, two important features are inherited:

1. The use of HTTP URIs provides a simple way to create unique global identifiers for services. Compared to, e.g., a universally unique identifier (UUID), Linked USDL URIs are more adequate to service distribution networks since they are managed locally by service providers following a process similar to the domain name system (DNS).
2. The same HTTP URI, which provides a global unique identifier for a service, also serves as endpoint to provide uniform data access to the service description. A Linked USDL URI can be used by, e.g., RDF browsers, RDF search engines, and web query agents looking for cloud service descriptions.

When a suitable Linked USDL HTTP URI has been selected for purchase (for example, our SugarCRM application), the customer can customize the service, for example, by selecting the pricing plan which is most suitable to his needs. Assuming that the `ultimate` plan is selected, the marketplace sends a service request for routing. The service includes the URI and an optional part (the customization string), separated by a question mark ("?"), that contains customization information. The syntax is a sequence of <`key`>=<`value`> pairs separated by a ampersand ("&"). Both, key and value, are URIs referencing

semantic concepts defined within the Linked USDL service description. For example, the URI `http://rdfs.genssiz.org/SugarCRM?pricePlan=pricing_SugarCRM_Ultimate` in which the key `pricePlan` and value `pricing_SugarCRM_Ultimate` are concepts defined within the Linked USDL description of the SugarCRM application (in this example, the full URI was omitted to make the notation more compact). The customization string adopts the same structure as query strings, a recommendation of the W3C.

4.3 Intelligent Routing of Service Requests

Based on the global service identification and description access, the SIOPP platform relies on a content-based routing [14] strategy to forward service requests, generated by service marketplaces, to TOSCA deployment providers. The routers examine the content of Linked USDL service descriptions, apply SPARQL queries and reasoning rules—providing some degree of intelligence within the router—to determine the providers who are able to provide the respective service. The mapping of Linked USDL URIs, pointing to an offering with the application provisioned by TOSCA, is realized by the distributed routing logic depicted in Figure 2. The proposed mechanism is designed with three routing layers: (i) the Global Routing Layer (GRL), (ii) the Local Routing Layer (LRL), (iii) and the TOSCA Routing Layer (TRL).

The Global Routing Layer uses a routing table to map Linked USDL URIs, describing the high level requirements for the application provisioning, such as pricing model, to providers which are able to provision the application accordingly. The GRL receives an USDL URI from a marketplace, looks up appropriate providers and selects one of them. This selection may take into consideration further conditions defined by the user such as pricing, payment method, or security requirements. However, these aspects are out of scope for this paper. Each provider is referenced by an endpoint implementing an interface used by the GRL to pass requests to the Local Routing Layer of the respective provider in order to trigger the provisioning of the application.

The Local Routing Layer uses the Linked USDL URI and a (local) routing table to select the corresponding TOSCA archive and TOSCA container, which brings us to the TOSCA Routing Layer. The installations are referenced by a TOSCA service id which can be used to trigger the provisioning of the service by the Local Routing Layer via the TOSCA-Runtime Environment. In addition, the routing table stores the input message used to invoke the build plan. This input message contains provider-specific information, for example, IP ranges or credentials, as well as field to pass the Linked USDL URI to the build plan. The plan may use the URI to configure the application based on the information represented by the URI or, in addition, may inspect the Linked USDL service description to gather more information, e.g., details of the selected price plan. Thus, the third TOSCA Routing Layer executes and configures the actual provisioning of the service.

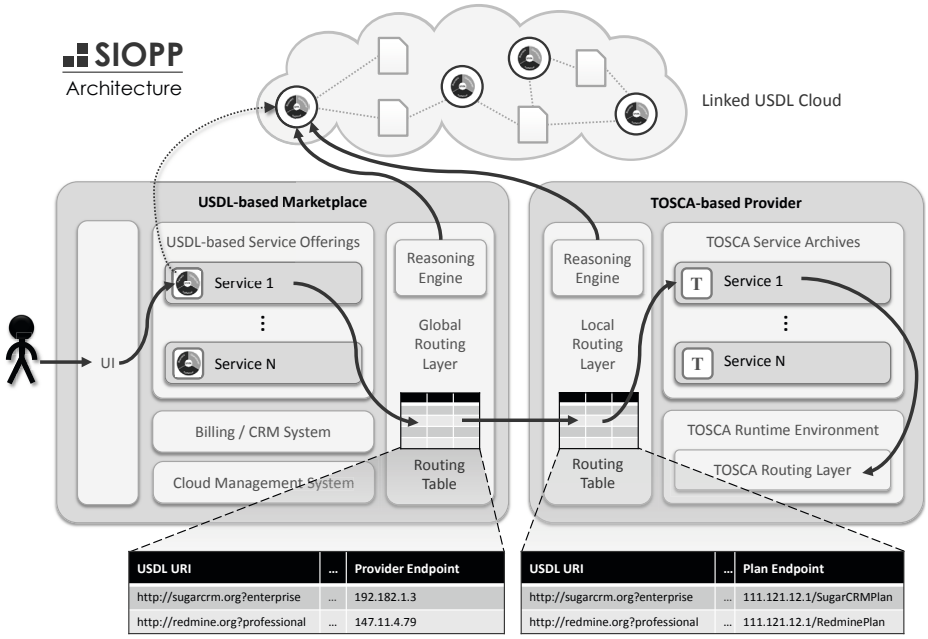

Fig. 2. Intelligent content-based routing mechanism of SIOPP

Listing 1.3 shows an example of an input message used by the build plan to deploy SugarCRM on Amazon EC2 (described in Section 3.4). The message contains credentials of the Amazon account to be used (line 2 and 3), the geographic region where the virtual machines should be located (line 4), and a pointer to the USDL offering (line 5). The USDL URI is used by the plan to query the Linked USDL offering by using SPARQL and adjust the deployment. In our prototype, deciding between the deployment options `enterprise` or `ultimate` is done based on the selected USDL pricing plan.

```
1 <BuildSugarCrmUltimateRequest>
2     <AmazonAccessKey>-key-</AmazonAccessKey>
3     <AmazonSecretKey>-secret-</AmazonSecretKey>
4     <EC2Endpoint>ec2.eu-west-1.amazonaws.com</EC2Endpoint>
5     <USDLURI>http://rdfs.genssiz.org/SugarCRM?pricePlan=
          pricing_SugarCRM_Ultimate</USDLURI>
6 </BuildSugarCrmUltimateRequest>
```

Listing 1.3. SugarCRM build plan input message

Listing 1.4 shows the SPARQL query used by the build plan to inquire about the options which are attached to the pricing plan included by the (customized) USDL URI. The options are then installed automatically.

```
1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3 PREFIX usdl: <http://www.linked-usdl.org/ns/usdl-core#>
4 PREFIX price: <http://www.linked-usdl.org/ns/usdl-pricing#>
5 select ?option
6 where {
7   pricePlan + price:hasPriceComponent ?priceComponent .
8   ?priceComponent price:isLinkedTo ?option . }
```

Listing 1.4. The SPARQL query issued by the build plan

The use of Linked USDL enables the content-based routing to be (1) intelligent and (2) adaptable. First, inference engines can be used to derive additional information not explicitly contained in a service description. For example, the Similarity Ontology (SO)[16] can be used to determine if the description of a service to be routed is `so:similar` or `so:related` to some service target. The use of transitive properties, such as `so:matches`, can be explored to infer implicit routing rules. Second, an adaptable content-based routing is achieved. It is possible to extend service descriptions with, for example, domain-dependent information as done in the field of logistics with the FInest project[8]; or enhance service descriptions with external information sources, for example, using dbpedia, YAGO, or freebase[9]. This contrasts to existing approaches which rely on closed schemas like WSDL. The routing mechanism works with the extensibility of Linked USDL and is able to process extended service descriptions. The modeling of our SaaS SugarCRM included domain-dependent vocabulary from the CRM field (see Section 3.2). Nonetheless, the evaluation of the impact of additional domain-dependent information on routing was out of scope for this paper.

4.4 Dynamic Binding of Deployment Descriptors

The binding of Linked USDL service offerings to TOSCA service deployments is done in a loosely coupled manner using TOSCA deployment descriptors. A TOSCA deployment descriptor is the combination of (i) a TOSCA Service Archive identifier, (ii) the endpoint of its build plan, and (iii) the respective input message for the build plan. The provider's TOSCA Runtime Environment is able to automatically process the TOSCA deployment descriptors stored in the routing table of the Local Routing Layer. The Local Routing Layer maps the USDL URIs, passed by the Global Routing Layer to the provider, to the corresponding TOSCA deployment descriptors solely based on this URI. Our approach uses the publish-subscribe pattern which enables providers to dynamically offer their provisioning capabilities to marketplaces. This design achieves advantages in resource management, workload distribution, and maintenance operations. For example, if a service instance is slowed down by a high request rate, the provider is able to instantiate and subscribe a second instance to distribute the workload.

[8] `http://finest-ppp.eu/`

[9] `http://dbpedia.org; www.mpi-inf.mpg.de/yago-naga; www.freebase.com`

5 Evaluation

An evaluation was conducted to assert the feasibility, performance, separation of concerns, and limitations of the prototype developed:

Feasibility. The integration of USDL and TOSCA required a loosely coupled platform to account for the dynamic nature of service advertisements and service provisioning. Three main challenges emerged during the construction of the SIOPP prototype: (i) global service identification and remote description access, (ii) intelligent routing of service requests, (iii) and dynamic binding of deployment descriptors. We were able to exploit USDL features (inherited from linked data principles) to achieve an unique service identification schema using Linked USDL URIs and a uniform data access [12] to service descriptions using Linked USDL HTTP URIs. In contrast to using, e.g., web APIs, it enabled a simpler integration of the marketplace and service providers' platforms responsible for service deployment and management. The use of a decentralized management of unique service identifiers was a scalable solution for the Internet of services. The use of SPARQL for the content-based routing [14] of service requests enabled a more flexible querying mechanism when compared, here again, with the access to web APIs to retrieve service data, since a full access to the service specifications is possible remotely. The dynamic association of a specific TOSCA deployment descriptor with a USDL service offering was achieved using a publish-subscribe pattern [15]. This enables cloud providers to quickly adapt to peak demand by distributing service requests to different TOSCA Runtime Environments. Compared to other approaches, e.g., which use business process management or integration by web services, the platform achieved a higher degree of decoupling, certainly more suitable for large scale deployments.

Performance. Regardless of using SIOPP or not, the application has to be setup using a build plan. Thus, we measured the performance of each component separately, to analyze the added runtime. For the GRL we used a hashtable with 500,000 entries and looked up 5,000 entries with a total lookup time of 3ms. To measure the LRL we used a hashtable with 10,000 entries and looked up 1,000 entries which resulted in a total lookup time of 2ms. The measurement setting was Win7-64bit, JRE 1.7, Intel i5-2410M, 2,3GHz. The build plan was adapted to return immediately after executing the SPARQL query, i.e., before the actual deployment at Amazon started, has an average runtime of 289ms ($\sigma = 76$). The runtime of the plan deploying SugarCRM varies between 4 and 7 minutes, depending on the provisioning time of the VMs at Amazon EC2. Thus, the overhead caused by SIOPP, even for peak demands, is negligible in our scenario.

Separation of Concerns. The distributed multi-layer routing logic enables the separation of concerns: The GRL reflects high level information, e.g., the global routing table may store information about the country of the provider for legal aspects. The LRL handles lower level aspects such as load balancing information, e.g., new service instances can be registered in the local routing table for peak

demands. The TRL enables, for example, implementing security aspects directly in management plans. This separation allows providers to focus on configuration and subscription and to design their own strategies based on individual aspects such as pricing. There is no need to understand the application's management.

Limitations. Since our routing approach has only three fixed routing components, it is not scalable for a global operation. One way to address this limitation is to adopt a peer-to-peer architecture using an overlay network organized with, e.g., the Simple Knowledge Organization System (SKOS). The network can be partitioned according to service domains (e.g., healthcare, finance, and logistics). Requests can be routed from domain to domain/subdomains linked using SKOS properties (e.g., `skos:narrower` and `skos:member`). The customization string (see Section 4.2), works well with simple customization. However, it is inadequate for condition-based based customization, i.e. if logical conditions need to be sent along with service requests. Also, associating USDL URIs with concrete input values for build plans has been found to be difficult if there is no description on how the values affect the deployment.

6 Related Work

While several researchers have studied different architectures for marketplaces (e.g., [17,18]), no known studies have been focused specifically on how cloud service offerings can be connected to their automated provisioning. Furthermore, except for a detailed study on cloud computing standardization efforts [19], research on interoperability between cloud standards has been overlooked. Our efforts to integrate service specifications and standards was first reported by Cardoso et al. [20]. We concluded that the use of model-driven approaches to transform models was too complex for large scale projects. Therefore, in this paper we based our approach on Linked USDL [6] to achieve a more loosely coupled and simpler alternative.

Pedrinaci et al. [21] propose the iServe platform to publish linked services, which is a subclass of Linked USDL services representing WSDL, RESTful, OWL-S, and WSMO services. Kirschnick et al. [22] reuse existing solutions to install and configure software to cloud environments. In both of these works, the question of how service offerings can trigger the remote deployment of a service was not addressed.

Jayasena et al. [23] integrate different financial standards, such as IFX and SWIFT, with an ontology to resolve semantic heterogeneity. This approach works well when the standards being integrated represent similar information. Cardoso et al. [24] follow a similar solution and add the notion of dynamic mappings to establish relations between different specifications. Nonetheless, both achieve limited results when overlap information is small, which is the case of USDL and TOSCA.

While these works use a bottom-up approach, other research took a top-down approach. For example, the Open Services for Lifecycle Collaboration (OSLC) [25] community created specifications to prescribe how tools (e.g., requirements tools, change management tools, testing tools, and so forth) should

be implemented and integrated to exchange data. While the approach has shown to be extremely successful, it cannot be applied to the problem we tackle since the specifications we integrate already exist and were developed by different organizations [19].

7 Conclusions

The emergence of cloud standards and specifications, such as USDL and TOSCA, brings the necessity to evaluate to which extend they are interoperable. In the presented approach we developed a prototypical platform to integrate both specifications by modeling the description and deployment of a commercial SaaS application: SugarCRM. The prototyping process enabled us to identify the challenges and limitations of making USDL and TOSCA interoperable. Important findings indicate that the use of a global service identification and description access enables a 'lightweight' integration without having the need to agree on proprietary web APIs. The multi-level and intelligent routing of service requests allows making routing decisions on different levels of granularity (e.g., legal, pricing, and security). The routing based on Linked USDL URIs achieves a high performance since analysis can be made, in many scenarios, only at the URI level. For a more advanced routing, Linked USDL descriptions can be remotely accessed. Finally, the dynamic binding of deployment descriptors with services enables providers to react to changing demands and workloads in a flexible manner.

Acknowledgment. This work was partially funded by the BMWi project CloudCycle (01MD11023). Additionally, we would like to thank to Alistair Barros for the interesting discussions during our research study.

References

1. Borenstein, N., Blake, J.: Cloud computing standards: Where's the beef? IEEE Internet Computing 15(3), 74–78 (2011)
2. Machado, G.S., Hausheer, D., Stiller, B.: Considerations on the interoperability of and between cloud computing standards. In: 27th Open Grid Forum (OGF27), G2C-Net Workshop: From Grid to Cloud Networks, OGF (October 2009)
3. Cardoso, J., Barros, A., May, N., Kylau, U.: Towards a unified service description language for the internet of services: Requirements and first developments. In: IEEE International Conference on Services Computing, Florida, USA (2010)
4. Barros, A., Oberle, D.: Handbook of Service Description: USDL and Its Methods. Springer (2012)
5. OASIS: Topology and Orchestration Specification for Cloud Applications Version 1.0. Working Draft 14 (November 2012)
6. Cardoso, J., Pedrinaci, C., Leidig, T., Rupino, P., Leenheer, P.D.: Open semantic service networks. In: The International Symposium on Services Science (ISSS 2012), Leipzig, Germany, pp. 1–15 (2012)
7. Cardoso, J., Winkler, M., Voigt, K.: A service description language for the internet of services. In: First International Symposium on Services Science (ISSS 2009), Leipzig, Germany (2009)

8. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. International Journal on Semantic Web and Information Systems 4(2), 1–22 (2009)
9. Binz, T., Breiter, G., Leymann, F., Spatzier, T.: Portable Cloud Services Using TOSCA. IEEE Internet Computing 16(03), 80–85 (2012)
10. Breitenbücher, U., Binz, T., Kopp, O., Leymann, F., Schumm, D.: Vino4TOSCA: A Visual Notation for Application Topologies based on TOSCA. In: Meersman, R., Panetto, H., Dillon, T., Rinderle-Ma, S., Dadam, P., Zhou, X., Pearson, S., Ferscha, A., Bergamaschi, S., Cruz, I.F. (eds.) OTM 2012, Part I. LNCS, vol. 7565, pp. 416–424. Springer, Heidelberg (2012)
11. Hors, A.L., Nally, M.: Using read/write Linked Data for Application Integration: Towards a Linked Data Basic Profile. In: Linked Data on the Web (2012)
12. Ziegler, P., Dittrich, K.: Three decades of data intecration – all problems solved? In: Jacquart, R. (ed.) Building the Information Society. IFIP, vol. 156, pp. 3–12. Springer, Boston (2004)
13. Bizer, C., Cyganiak, R., Gauss, T.: The RDF book mashup: From web apis to a web of data. In: Proceedings of the ESWC 2007 Workshop on Scripting for the Semantic Web, vol. 248 (2007)
14. Carzaniga, A., Rutherford, M.J., Wolf, A.L.: A routing scheme for content-based networking. In: Proceedings of IEEE INFOCOM 2004, Hong Kong, China (2004)
15. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley, Boston (2003)
16. Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl:sameAs isn't the same: An analysis of identity in linked data. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 305–320. Springer, Heidelberg (2010)
17. Menychtas, A., Gomez, S.G., Giessmann, A., Gatzioura, A., Stanoevska, K., Vogel, J., Moulos, V.: A marketplace framework for trading cloud-based services. In: Vanmechelen, K., Altmann, J., Rana, O.F. (eds.) GECON 2011. LNCS, vol. 7150, pp. 76–89. Springer, Heidelberg (2012)
18. Akolkar, R., Chefalas, T., Laredo, J., Peng, C.S., Sailer, A., Schaffa, F., Silva-Lepe, I., Tao, T.: The future of service marketplaces in the cloud. In: IEEE Eighth World Congress on Services, pp. 262–269 (2012)
19. BMWi: The standardisation environment for cloud computing. Technical report, Germany Federal Ministry of Economics and Technology (Febraury 2012)
20. Cardoso, J., Voigt, K., Winkler, M.: Service engineering for the internet of services. In: Filipe, J., Cordeiro, J. (eds.) ICEIS 2008. LNBIP, vol. 19, pp. 15–27. Springer, Heidelberg (2009)
21. Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J., Domingue, J.: iServe: a linked services publishing platform. In: Ontology Repositories and Editors for the Semantic Web Workshop. CEUR Workshop Proceeding, vol. 596 (2010)
22. Kirschnick, J., Alcaraz Calero, J.: Toward an architecture for the automated provisioning of cloud services. IEEE Communications Magazine 48(12), 124–131 (2010)
23. Jayasena, S., Bressan, S., Madnick, S.: Financial information mediation: A case study of standards integration for electronic bill presentment and payment using the coin mediation technology. In: Shan, M.-C., Dayal, U., Hsu, M. (eds.) TES 2004. LNCS, vol. 3324, pp. 152–169. Springer, Heidelberg (2005)
24. Cardoso, J., Bussler, C.: Mapping between heterogeneous XML and OWL transaction representations in B2B integration. Data & Knowledge Engineering 70(12), 1046–1069 (2011)
25. OSLC Core Specification Workgroup: OSLC core specification version 2.0. Technical report, Open Services for Lifecycle Collaboration (August 2010)