A Versatile and Scalable Everything-as-a-Service Registry and Discovery

Josef Spillner and Alexander Schill
Faculty of Computer Science, Technische Universität Dresden, 01062 Dresden, Germany

Keywords: Service Registry, Service Model, Service Description, Matchmaking.

Abstract: Networked humans and autonomous systems rely increasingly on service-oriented architectures. Conventional systems often focus on digitally delivered web services or on business services with varying degrees of formalisation. According to service integration principles, the service registry as the pivotal point between providers and consumers determines the types of services which can be found and delivered. With the growing importance of cloud computing topics in research and industry, more generic service models, description languages and registry architectures are proposed to extend the breadth towards truly reusable Everything-as-a-Service (XaaS) registration entries. Despite the proposals, there is a shortage of capable and extensible XaaS registries. In this paper, we present such a system and share our experience in using it for service-related experiments.

1 INTRODUCTION

Everything-as-a-Service (XaaS) describes a spectrum of functionality which can be uniformly described, negotiated and brokered through uniform means, and executed and delivered through non-uniform ways. The term has been increasingly popular in service-oriented architectures, cloud computing and service science circles (Banerjee et al., 2011; Didoné and de Queiroz, 2011; Ferrario et al., 2011). It encompasses several delivery-related service classes. In particular, software- and hardware-provided web services, data and human services, as well as further emerging service classes such as gamification or electricity services which are all somehow incorporated into or orchestrated by information systems. Beside the service class, each service belongs to a certain functional domain. Hotel booking, e-car charging, text mining, data storage and many more services can thus be expressed as a combination of service class and service domain. Finally, each service of a particular combination can be distinguished with non-functional properties. Service pricing, location, quality, safety and security are among the vast possible properties.

Researchers have been concerned with service selection and substitution problems based on required properties or dynamic composition based on service functions and coincidence for some time (Cavalcante et al., 2012). In order to find the best suitable service for a certain demand, service registries have been established as pivotal systems between providers and consumers. Based on those, service offering and discovery workflows have been proposed to guide the user from having a service idea to offering a service and from expressing the demand to achieving the desired result through matchmaking, respectively (Guinard et al., 2010). Yet, despite having existed for a long time, there are several shortcomings in how the current generation of registries work, scale and fit the XaaS spectrum.

In this paper, we argue analytically about the shortcomings in existing service models and description languages, registration models and registry and discovery systems. We then use this knowledge to derive a suitable problem statement and present our solution approach which results in a versatile and scalable XaaS registry. Finally, we evaluate this registry experimentally with thousands of imported services from various domains and service classes.

2 EXISTING WORK

Previous research about service registration and discovery focuses on three disjoint and yet closely related topics: description languages, registration models, and realised registry and discovery systems. The state of the art in these topics will be summarised in the following paragraphs. For extensive reviews and surveys, we refer to (D'Mello and S., 2010) and (Nair and Gopalakrishna, 2010).

Spillner J. and Schill A..
A Versatile and Scalable Everything-as-a-Service Registry and Discovery.
DOI: 10.5220/0004373301750183
In *Proceedings of the 3rd International Conference on Cloud Computing and Services Science* (CLOSER-2013), pages 175-183
ISBN: 978-989-8565-52-5

2.1 Existing Description Languages

A service description contains valuable information about a service. It is produced by the service provider and inspected by prospective and actual service consumers. Therefore, it needs to be uniform, precise, correct and complete (D'Mello and S., 2010). Typical descriptions convey information about the service class and domain, interfaces to access the service, the behaviour along with input and output parameters, preconditions and effects (IOPEs), as well as arbitrary non-functional properties about pricing, service quality (QoS), security and other distinguishing factors.

Table 1 gives a comprehensive overview about approaches to describe services with various domain and technology assumptions. Not included are complementary languages to describe message structures, agreements, guarantees and policies about the service usage. The language characteristics encompass completely domain-independent services (e.g. OWL-S (Solanki et al., 2004), WSML (de Bruijn et al., 2006), USDL (Barros et al., 2012)), domain-independent services with a restriction to a certain interaction paradigm (e.g. WSDL and WADL (Pautasso et al., 2008), RIDDL (Mangler et al., 2009)) as well as mostly domain-dependent ones, with permissible subdomains for specialisation, without any specific requirement on the format of the interaction (e.g. MSDL, IDNL (Ghijsen et al., 2012), CoCoOn (Zhang et al., 2012), DEMODS (Vu et al., 2012) and the Text Mining Service Ontology (Pfeifer and Schill, 2012)). OCCI is a special case because it mandates a certain RESTful protocol to obtain the descriptive elements and is hence restricted to both a domain and an interaction protocol. The description languages have a varying degree of suitability for any given use case, maturity and adoption. Hence, it becomes clear that a generic XaaS registry should accomodate arbitrary combinations of instances of these languages.

Table 1: Overview about service description languages.

Acronym	Name	Service Types	Adoption
WSDL	Web Service Description Language	SOAP and REST web services	huge
WADL	Web Application Description Language	REST web services	academic
RIDDL	RESTful Interface Definition and Declaration Language	REST web services	academic
USDL	Unified Service Description Language	web and human services	small
OWL-S	Web Ontology Language for Services	generic	academic
WSML	Web Service Modelling Language	generic	academic
OCCI	Open Cloud Computing Interface	infrastructure services	small
INDL	Infrastructure and Network Description Language	infrastructure services	academic
CoCoOn	Cloud Computing Ontology	infrastructure services	academic
MSDL	Mobilis Service Description Language	collaboration services	academic
DEMODS	Description Model for Data as a Service	dataset services	academic
–	Text Mining Service Ontology	information services	academic

2.2 Existing Registration Models

Registration models define how declarative service artefacts are mapped onto the structures present in service registries. In particular, they mandate the acceptable artefacts, certain granularities – e.g. full or partial updates – and cardinalities of documents per registration entry. Universal Description, Discovery and Integration (UDDI) is a service registry specification which mandates both a registration model and a registry system with web service interfaces. The registration model encompasses technical models, the so-called *tModels*, to represent unique concepts (Paolucci et al., 2002), along with industry taxonomies and enterprise models. Models are shared between entries of the same type of service. For instance, a WSDL description would be split into its abstract (interface description) and concrete (provisioning) parts, with only the former relating to the model and the latter being mapped onto a *businessService* entry. Going beyond web services, the ebXML Registry Information Model (Hofreiter et al., 2002) controls which content and metadata types can be registered in an appropriate registry. Each entry is called a `RepositoryItem` whereas metadata on it is called `RepositoryObject`. While the UDDI model is focusing on web services as a subset of XaaS, and ebXML manages content as a superset of XaaS artefacts, the Dragon registration model supports several service-related documents. Specifically, WS-Agreement SLA documents can be registered alongside informative WSDL descriptions. Further artefact types such as policies are not supported.

Table 2 summarises our analysis of registration models. It becomes obvious that the specialisation of Dragon is too limited to cater for new developments (e.g. security policies) whereas the generalisation of UDDI and ebXML makes it ambiguous how exactly multi-artefact services are to be registered.

Table 2: Overview about service registry models.

Name	Service Descriptions	Characteristics
UDDI	various through tModels	focus on web services
ebXML	any	for any content
Dragon	WSDL, WS-Agreement	focus on web services

2.3 Existing Registry Systems

Architectures and software systems to register and find services are layered as follows: the storage area (repository) to persist submitted service descriptions, the programmatic interface (registry) to perform the registration and retrieval operations, and the user-centric interface (marketplace) which appears as broker, portal or other user interface to let the user perform the offering and discovery processes. The systems can further be classified as centralised, decentralised (peer-to-peer) or federated architectures. Some are freely available or have hosted instances. Our analysis compares representative systems without the claim for completeness given the multitude of systems available.

UDDI and ebXML registries implement the aforementioned UDDI and ebXML registration models. Apache jUDDI and OpenUDDI are open-source implementations of UDDI. Commercial UDDI products are available from many companies, including Oracle Service Registry, Novell UDDI Server and SAP NetWeaver. While the former UDDI-based Universal Business Registry (UBR) has not survived, some publicly operated registries still support the UDDI specification, including `xmethods.net`. UDDI registries have been found to not be a sufficient basis for dynamic service selection. The reasons include the lack of a sophisticated full-text search for unstructured content, the lack of support for non-functional properties, the assumption about the services being offered by an enterprise and the asymmetric requirements specification, among others (Field and Hoffner, 2002). Nevertheless, researchers are attracted by its standard nature and have proposed distributed UDDI flavours such as Ad-UDDI (Du et al., 2005).

The USDL Marketplace contains a registry dedicated to the exchange of USDL artefacts. It is business-oriented in a way that it recommends services for business scenarios and allows for business value network creation based on contracts and subcontract relationships. The marketplace is built as a J2EE application with various Java frameworks (Hibernate, Spring, Seam) connected to a SQL database. It has been designed taking into account experiences from previous USDL registries such as Agora (Cardoso et al., 2010). There is no known public installation. The Membrane SOA Registry and the Service-Finder are registries dedicated to WSDL-described services. Both are active registries which monitor the service availability. The Service-Finder even actively crawls the web for additional descriptions and meta-data to enrich the entries (Steinmetz et al., 2009). Public instances with convenient human invocation interfaces run at `service-repository.com` and `webservices.seekda.com`, respectively. The IBM Dynamic Matchmaking Engine has support for functional and non-functional properties which can be either fixed or dynamic so that their value will be determined in the procurement process (Field and Hoffner, 2002). As such, this registry takes over contract negotiation tasks which in other architectures are performed in separate components. Depot (Abu-Jarour, 2010) has been proposed to increase the quality of service descriptions in a registry. It pro-actively crawls information sources for new descriptions similar to the Service-Finder.

A fully decentralised architecture based on peer-to-peer hashtables and gradient topologies has been proposed to keep deployment costs low and to handle high scalability (Sacha et al., 2007).

All architectures and software systems for the registration and discovery of services are summarised in Table 3. The variety regarding the status and free availability of the implementation, the existence of a hosted instance and the architectural choices is remarkably high.

Table 3: Overview about service registries.

Name	Characteristics
jUDDI	centralised; available
Ad-UDDI	decentralised; architecture proposal; monitoring
USDL Marketplace	centralised; available
Membrane SOA Registry	centralised; available; hosted
Service-Finder	centralised; proprietary; hosted; crawling
IBM Dynamic MME	centralised; proprietary
Depot	centralised; proprietary; crawling
P2P Registry	decentralised; architecture proposal

2.4 Problem Definition

Existing registration models and registry systems are inadequate for the exchange of information about XaaS. All of the analysed models were too specific or too general regarding the possible and the customary declarative service artefact types. The analysis of registries reveals a lack of easily usable, freely available, versatile, extensible and scalable systems: Just three out of the analysed eight systems are freely available.

Out of the three (jUDDI, USDL Marketplace, Membrane SOA Registry), only one (the USDL Marketplace) targets versatile services ranging from human to web services. However, its extensibility is limited by the highly capable and yet quite complex service model, its scalability is unknown and its usability is severely restricted by the lack of easily installable software packages. Furthermore, it doesn't ship with service domain definitions. Today's requirements of research projects and pilot studies in enterprise service and cloud environments are thus best fulfilled with a novel kind of XaaS registry beyond the scope of just web services.

3 SOLUTION APPROACH

We propose a three-concerns solution consisting of an appropriate open and extensible description language for services covering as many XaaS domains as possible, a registration model and a system to perform the registration and subsequent service discovery.

3.1 Description Model and Language

Description languages for services capture all aspects of fundamental service models. The service classes and domains shown in Figure 1 represent both fully automated virtual and cyber-physical services as well as human services. A requirement for a truly versatile XaaS registry which operates in an open Internet of Services, and hence a requirement for a corresponding description language, is to unify these services as much as possible while restricting the domain-specific service characteristics as little as possible.

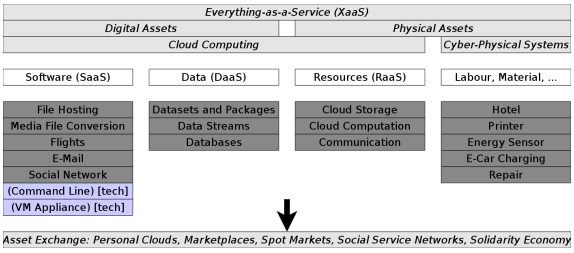

Figure 1: Scope of Everything-as-a-Service (XaaS) in an Internet of Services.

Following the analysis of existing service description languages, we conclude that semantic languages are very suitable for XaaS due to their separation of language and extensible vocabulary. On the other hand, the lack of vocabulary in the languages themselves leads to the requirement of introducing a specific catalogue of vocabulary and concepts on top of the languages. We have chosen WSML as the unintrusive modelling language of choice due to its strong support for non-functional properties and general service orientation. WSML is a syntactical representation of the Web Service Modelling Ontology (WSMO) (de Bruijn et al., 2006). Hence, we propose a specific WSML ontology catalogue, called WSMO for the Internet of Services - WSMO4IoS, as XaaS vocabulary collection which covers both reusable domain-independent base concepts and domain-specific concepts. WSML can be transformed to OWL so that the modelling and reasoning support among the semantic service tools should be high enough for all use cases. Figure 2 demonstrates the scope of WSMO4IoS for both base and domain ontologies. The term unintrusive refers to the fact that a service doesn't need to be described in WSML prior to the registration into our registry. In such a case, appropriate minimal descriptions can be generated automatically based on information extracted from the present artefacts.

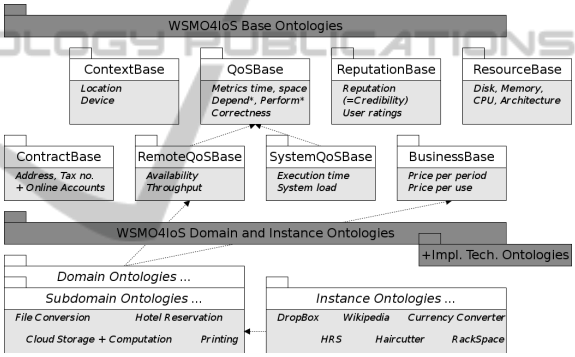

Figure 2: WSMO4IoS: Ontologies for the Internet of Services and Everything-as-a-Service

WSMO4IoS is maintained as a growing versioned catalogue of both base and domain ontologies as well as corresponding imaginary and real instance services. The catalogue is made available publicly alongside its specification document (Spillner, 2013). The subsequent sections assume the use of the January 2013 version with 10 base and 13 domain ontologies. The domain-specific modelling complexity ranges from two concepts for social services to nine concepts for Cloud storage services. The latter one is described in detail in (Spillner et al., 2013).

3.2 Registration Model

The Tradeable Services Model introduces the notion of logical service packages which can be exchanged and traded through open markets. Service providers and potential consumers meet through these markets for search and subsequent matchmaking. Registries

as core elements of markets support the search and matchmaking with registration models which allow for a precise evaluation of service functionality and non-functional properties. In the Tradeable Services Model, declarative service description artefacts are combined with executable implementation artefacts. The implementation artefacts are however considered external to the registry and typically would be stored in a service execution container or repository, whereas the description artefacts are stored by the registry itself. Figure 3 shows one possible structure of a service package. It can be seen that services can be complex entities with heterogeneous descriptions of different cardinality. Some descriptions are related to others.

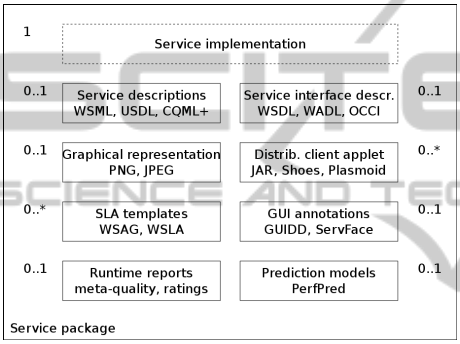

Figure 3: Service package according to the Tradeable Services Model.

A generic registry must therefore accomodate certain hierarchies. For instance, web services often mandate different technical interfaces, ports and protocols, whereas business services often can be delivered with varying guarantees based on the same technical implementation. Figure 4 proposes a two-level artefact registration model where both on the top level and within each interface descriptive artefacts can be registered. There can also be references between the artefacts themselves, for instance a SAWSDL reference from a WSDL file to an ontology.

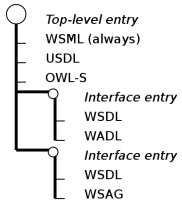

Figure 4: Structure of entries in the registry.

No strong assumption is made about the artefact types. The model allows for generic artefacts of either single or multiple occurrence at each level. It is the registry's concern to govern which artefact types can be registered with which multiplicity. Table 4 contains a suitable configuration which has developed over time in several research projects with our involvement. It should be noted that even in the case of single occurrence, the artefact type may itself be structured in a way that multiple entries be present. For example, an Agents description lists multiple software agents which autonomic service integration systems use to interact with the service.

Table 4: Tradeable Services Model permissible artefact configuration.

Artefact Type	Multiplicity
WSDL service description	1
WADL service description	1
USDL service description	1
RDF resource description	1
UISDL user interface service	1
GUIDD deployment descriptor	1
WSAG agreement template	multiple
Applet for interaction	multiple
Clients descriptor	1
Agents descriptor	1
Source/Implementation package	1
Manual page	1
Resource locator	1
Dependency reference	1
Icon	1

3.3 Registry and Discovery System

Service registries should themselves be services in order to streamline their integration into service-oriented landscapes and the Internet of Services. In terms of complete service platforms built around registries, each platform component acts as a platform service with multiple role-specific programmatic and interactive interfaces. Typical roles are service providers, consumers and platform operators, among others. The ConQo semantic service registry is such a platform service. It has initially been proposed to facilitate the discovery of context- and quality-tagged web services (Stoyanova et al., 2008). We have fully implemented and extended ConQo significantly towards a generalisation regarding the service classes, functional service domains and non-functional properties. All properties can be dynamically updated through a monitoring submission endpoint (ConQoMon) beyond the coarse-grained updates of complete service artefacts. This evolution has turned ConQo into a versatile XaaS registry whose behaviour is controlled by the deployed artefact configuration and base and domain ontologies. Furthermore, we have refactored the code and substituted some algorithms for a much higher scalability. In particular, the former purely centralised ar-

chitecture has been extended into a passively master-slave-replicable/federated architecture in which entries from several registries can be downstreamed into another instance. Two WS-* extensions to ConQo's SOAP interface have been developed for this purpose: WS-DiffTransfer allows for a differential transfer of only modified entries instead of all entries, and WS-Paging adds a pagination similar to browsing results known from web search sites. The evaluation section refers to this current version of ConQo.

Figure 5 outlines the high-level architecture of the registry components, the artefact repository, and the web service interfaces to the registry. ConQo is not a pro-active (crawling, self-updating) registry like Service-Finder or Depot. However, additional tools exist which make use of ConQo's interfaces to achieve an equivalent functionality. In particular, a service crawler registers services and a monitor supervises them (Spillner, 2010).

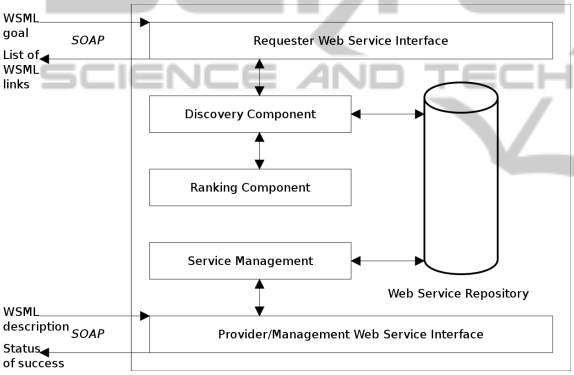

Figure 5: Architecture of the ConQo registry and discovery.

ConQo has several unique features not found in other registries, such as the ability to save searches for subsequent reverse auctions of providers who want to more precisely target the needs of the prospective service consumers, and the ability to assign reputation scores to the advertised service features. The search can be initiated from both full-text queries and descending domain browsing, with refinements through subdomain and non-functional property filters. An selected overview of the role-specific service interfaces to perform discoveries, rate services and update non-functional properties is given in Table 5. In total, there are 28 web service methods. Access to the methods is not protected, because this concern is better handled by platform services which authenticate and redirect requests.

Table 5: Some of the web service methods offered by ConQo.

Method Name and Signature	Role
initMatchmaker()	Admin
getAllMonitors()	Admin
addServiceComplete(wsml,img,user)	Provider
addInterfaceDocument(iri,doctype,doc)	Provider
getSearches()	Provider
getWebServices(domain)	Consumer
getInterfaceDocuments(iri)	Consumer
getReputationParameters(iri)	Consumer
achieveGoalText(wsml)	Consumer

4 EVALUATION

We evaluate our work by first presenting the resulting registry system as implementation of the proposed architecture and an ecosystem of tools around it. Then, we look at integration perspectives, measure the system's scalability and take a look at its usability.

ConQo has been implemented as a Java Servlet, called Matchmaker, which runs in an application container and offers service interfaces for graphical user interfaces. It uses the DIP QoS-Enabled Service Discovery Component as described in (Stoyanova et al., 2008), the WSML2Reasoner framework with a choice of reasoning engines, WSMO4J to parse WSML descriptions and a JDBC connector to a relational database such as MySQL for storing all service description concepts, service instance entries and artefacts. The registry is made available as open source software from http://serviceplatform.org to promote its wide use in academic and industrial research.

Over time, the ConQo matchmaker has been used in a number of scenarios and hence been integrated into various marketplaces and tools. Figure 6 summarises these efforts. The ConQo Cockpit is the native web interface for service discovery based on functional domain selection and subsequent non-functional property filtering. Specialised web interfaces exist for scenarios like application-like service stores and user interface services. Social service networks such as Servomat (Spillner, 2011) and Crowdserving (Spillner, 2010) combine service trading with professional service offers in social networks. A desktop data provider enhances the KDE desktop environment with services in addition to local applications in the user's application selection menu. Finally, command line applications exist to interact with the matchmaker and to register both WSML descriptions and other artefacts which are converted to WSML based on templates. For experience re-

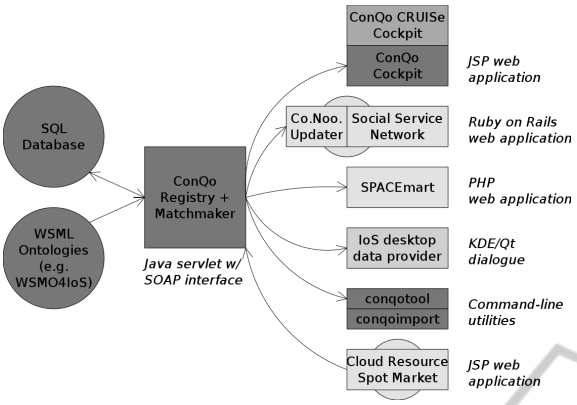

Figure 6: Usage and integration scenarios for the ConQo service registry.

ports about combinations of the ConQo matchmaker with its interfaces, we refer to the publications of the BMWi THESEUS/TEXO, BMBF CRUISe and EU-/ESF/SAB FlexCloud research projects which have contributed significant feedback for the continuous development.

In order to determine the scalability and suitability of ConQo for XaaS, it has been evaluated in experiments which involve thousands of existing and artificially generated service descriptions. Figure 7 shows the effect of the differential synchronisation extension (WS-DiffTransfer). Compared to the reference baseline of synchronising the entire contents of one ConQo instance into another one, WS-DiffTransfer causes a $20X$ overhead in the worst case, which is quickly offset by both a high number of entries and a low number of modifications. In most practical cases, there are even extreme performance gains due to the near-$O(1)$ complexity of WS-DiffTransfer. The implementation improvements are summarised in Figure 8. One of the most-used methods, the parametrised discovery entry point `getWebServicesFiltered()`, has been improved to a point where a ConQo instance with 4000 entries only requires $2s$ instead of

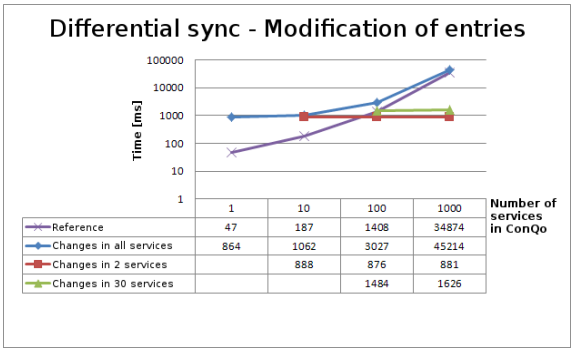

Figure 7: Scalability measured during a differential master-slave transfer of modified entries.

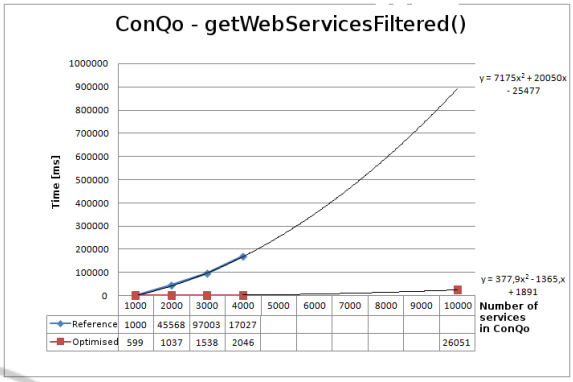

Figure 8: Increased scalability after combined refactoring and code improvements.

$17s$ and higher service counts such as 10000 become even possible with tolerable timing.

Finally, we have integrated the optimised registry into an experiment in which about 30000 service descriptions were collected from catalogues and crawlers, inserted into a scientific data repository, and injected into ConQo (Spillner, 2010). Given that 30000 is about the upper bound of unique publicly accessible services, this stresses the registry's scalability and fitness for real-world use cases.

The usability of ConQo is best demonstrated through screenshots of its user interfaces. Figures 9, 10 and 11 contain screenshots of the administrative service listing, discovery result and service browsing, respectively. The first two screenshots are taken from the ConQo Cockpit, whereas the third one is a native desktop dialogue on KDE connected to a remote ConCo instance. In both clients, selected services can be either just used or, in case a service level agreement template is found, used with contract protection after a negotiation in case the service platform supports this capability.

5 CONCLUSIONS

Service registries are essential pivotal elements in service-oriented architectures. With a clear trend towards marketplaces in an Internet of Services and Everything-as-a-Service (XaaS), the existence of versatile, scalable and yet simple registries becomes crucial. Our research motivates a three-concerns decoupling of registry systems implemented in software, abstract registration models supported by them, and concrete service description artefacts bound to registration entries following the models. The registry ConQo, with its role-oriented service interfaces and efficient query and synchronisation mechanisms,

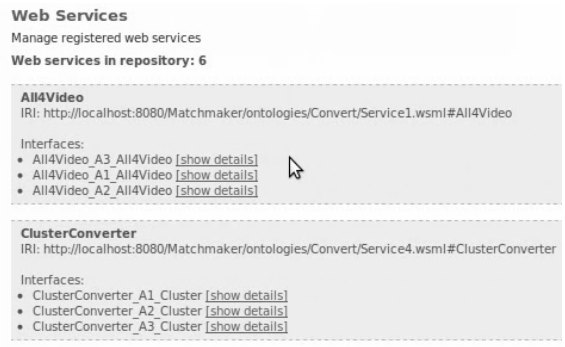

Figure 9: Screenshot of a service listing in the ConQo Cockpit administration menu.

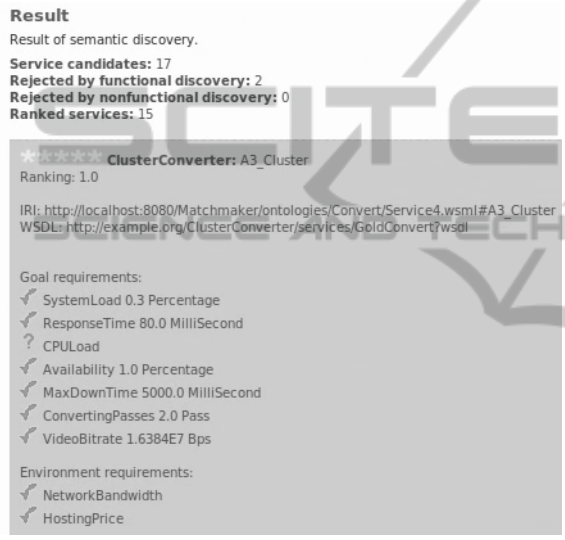

Figure 10: Screenshot of the weighted result listing following a query in the ConQo Cockpit discovery menu.

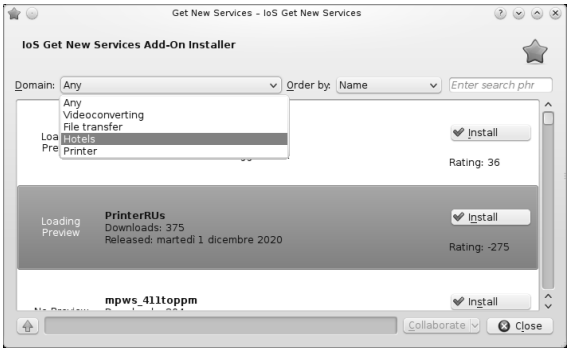

Figure 11: Screenshot of the KDE Get Hot New Services dialogue.

avoids the overhead and complexity of fully distributed registries while still achieving suitable scalability. Our future work concentrates on the specification of additional domain ontologies to accomodate more XaaS use cases.

ACKNOWLEDGEMENTS

This work has received funding under project number 080949277 by means of the European Regional Development Fund (ERDF), the European Social Fund (ESF) and the German Free State of Saxony. Many ConQo improvements have been contributed by Thomas Muckwar and Aleksander Heimrath based on the initial implementation by Bastian Buder.

REFERENCES

AbuJarour, M. (2010). A Proactive Service Registry With Enriched Service Descriptions. 9th Retreat of the HPI Research School. Neuruppin, Germany.

Banerjee, P., Bash, C., Friedrich, R., Goldsack, P., Huberman, B. A., Manley, J., Patel, C., Ranganathan, P., and Veitch, A. (2011). Everything as a Service: Powering the New Information Economy. *Computer*, 44(3):36–43.

Barros, A., Oberle, D., Kylau, U., and Heinzl, S. (2012). An Overview of USDL. In Barros, A. and Oberle, D., editors, *Handbook of Service Description - USDL and its Methods*, pages 185–215. Springer Berlin Heidelberg.

Cardoso, J., Barros, A., May, N., and Kylau, U. (2010). Towards a Unified Service Description Language for the Internet of Services: Requirements and First Developments. In *Proceedings of the IEEE International Conference on Services Computing (SCC)*, pages 602–609. Miami, Florida, USA.

Cavalcante, E., Batista, T., Lopes, F., Rodriguez, N., de Moura, A. L., Delicato, F. C., Pires, P. F., and Mendes, R. (2012). Optimizing Services Selection in a Cloud Multiplatform Scenario. In *1st Latin American Conference on Cloud Computing and Communications (LatinCloud)*, pages 31–36. Porto Alegre, Brazil.

de Bruijn, J., Lausen, H., Polleres, A., and Fensel, D. (2006). The Web Service Modeling Language WSML: An Overview. In *Proceedings of the 3rd European Conference on The Semantic Web: Research and Applications (ESWC)*, pages 590–604. Budva, Montenegro.

Didoné, D. and de Queiroz, R. J. G. B. (2011). Forensic as a Service - FaaS. In *Proceedings of the Sixth International Conference on Forensic Computer Science (ICoFCS)*, pages 202–210. Florianópolis, Brazil.

D'Mello, D. A. and S., A. V. (2010). A Review of Dynamic Web Service Description and Discovery Techniques. In *First International Conference on Integrated Intelligent Computing (ICIIC)*, pages 246–251. Bangalore, India.

Du, Z., Huai, J., and Liu, Y. (2005). Ad-UDDI: An Active and Distributed Service Registry. In *Proceedings of the 6th International Conference on Technologies for E-Services (TES)*, volume 3811 of *LNCS*, pages 58–71. Trondheim, Norway.

Ferrario, R., Guarino, N., Janiesch, C., Kiemes, T., Oberle, D., and Probst, F. (2011). Towards an Ontological

Foundation of Services Science: The General Service Model. In *10. Internationale Tagung Wirtschaftsinformatik*, page P47. Zurich, Switzerland.

Field, S. and Hoffner, Y. (2002). In search of the right partner. In *Collaborative Business Ecosystems and Virtual Enterprises*, chapter 7, pages 55–62. Kluwer.

Ghijsen, M., van der Ham, J., Grosso, P., and de Laat, C. (2012). Towards an Infrastructure Description Language for Modeling Computing Infrastructures. In *10th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, pages 207–214. Madrid, Spain.

Guinard, D., Trifa, V., Karnouskos, S., Spieß, P., and Savio, D. (2010). Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. *IEEE Transactions on Services Computing*, 3(3):223–235.

Hofreiter, B., Huemer, C., and Klas, W. (2002). ebXML: Status, Research Issues, and Obstacles. In *Proc. of 12th International Workshop on Research Issues on Data Engineering (RIDE)*, pages 7–16. San Jose, California, USA.

Mangler, J., Schikuta, E., and Witzany, C. (2009). Quo Vadis Interface Definition Languages? Towards a Interface Definition Language For RESTful Services. In *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 1–4. Taipei, Taiwan.

Nair, M. K. and Gopalakrishna, D. V. (2010). Look Before You Leap: A Survey of Web Service Discovery. *International Journal of Computer Applications*, 7(5).

Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. (2002). Importing the Semantic Web in UDDI. In *Web Services, e-Business and the Semantic Web*, volume 2512 of *LNCS*, pages 225–236. Toronto, Canada.

Pautasso, C., Zimmermann, O., and Leymann, F. (2008). Restful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. In *Proceedings of the 17th international conference on World Wide Web (WWW)*, pages 805–814. Beijing, China.

Pfeifer, K. and Schill, A. (2012). Semantic Description of Text Mining Services. In *Second International Conference on Advances in Information Mining and Management (IMMM)*. Venice, Italy.

Sacha, J., Biskupski, B., Dahlem, D., Cunningham, R., Dowling, J., and Meier, R. (2007). A Service-Oriented Peer-to-Peer Architecture for a Digital Ecosystem. In *2007 Inaugural IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, pages 205–210. Cairns, Australia.

Solanki, M., Martin, D., Paolucci, M., McIlraith, S., Burstein, M., McDermott, D., McGuinness, D., Parsia, B., Payne, T., Sabou, M., Solanki, M., Srinivasan, N., , and Sycara, K. (2004). Bringing Semantics to Web Services: The OWL-S Approach. In *First International Workshop on Semantic Web Services and Web Process Composition (SWSWPC)*, volume 3387 of *LNCS*, pages 26–42. San Diego, California, USA.

Spillner, J. (2010). *Methodik und Referenzarchitektur zur inkrementellen Verbesserung der Metaqualität einer vertragsgebundenen, heterogenen und verteilten Dienstausführung*. PhD thesis, Technische Universität Dresden. Faculty of Computer Science.

Spillner, J. (2011). An Environment for Educational Service Communities. *International Research Journal of Telecommunication Sciences*, 2(2):22–27.

Spillner, J. (2013). WSMO4IoS Specification and Ontology Catalogue. Online: http://serviceplatform.org/ spec/ wsmo4ios/.

Spillner, J., Müller, J., and Schill, A. (2013). Creating Optimal Cloud Storage Systems. *Future Generation Computer Systems*, 29(4):1062–1072. DOI: http://dx.doi.org/10.1016/j.future.2012.06.004.

Steinmetz, N., Lausen, H., and Brunner, M. (2009). Web Service Search on Large Scale. In *Service-Oriented Computing: 7th International Joint Conference, ICSOC-ServiceWave*, volume 5900 of *LNCS*, pages 437–444. Stockholm, Sweden.

Stoyanova, G., Buder, B., Strunk, A., and Braun, I. (2008). ConQo – A Context- and QoS-Aware Service Discovery. In *Proceedings of IADIS International Conference WWW/Internet*. Freiburg, Germany.

Vu, Q. H., Pham, T.-V., Truong, H.-L., Dustdar, S., and Asal, R. (2012). DEMODS: A Description Model for Data-as-a-Service. In *Proceedings of the 26th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pages 605–612. Fukuoka, Japan.

Zhang, M., Ranjan, R., Haller, A., Georgakopoulos, D., Menzel, M., and Nepal, S. (2012). An Ontology based System for Cloud Infrastructure Services Discovery. In *8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. Pittsburgh, Pennsylvania, USA.