# Telecommunication mashups using RESTful services

Alistair Duke[1], Sandra Stincic[1], John Davies[1], Guillermo Álvaro Rey[2], Carlos Pedrinaci[3], Maria Maleshkova[3], John Domingue[3], Dong Liu[3], Freddy Lecue[4] and Nikolay Mehandjiev[4]

[1]BT Innovate & Design, British Telecommunications plc. Ipswich, UK
alistair.duke, sandra.stincic, john.nj.davies}@bt.com
[2]iSOCO, Madrid, Spain
galvaro@isoco.com
[3]Knowledge Media Institute, The Open University, Milton Keynes, UK
c.pedrinaci, m.maleshkova, j.b.domingue, d.liu}@open.ac.uk
[4]University of Manchester, Manchester M15 6PB, UK
f.lecue@mbs.ac.uk, nikolay.mehandjiev@manchester.ac.uk

**Abstract.** Evolution in the telecommunications sector has led to companies within it providing APIs for their products and services, allowing others to build communication services into their own service offerings. In order to support mass adoption of this new approach, consumers of these APIs (many of which are RESTful) must be supported by a reduction in the complexity involved with describing, finding, composing and invoking them. Existing efforts to provide automation have, in general, focused on WSDL services rather than REST services. The paper explores the approach of the SOA4All project in supporting interaction with REST services which is being applied in a telecommunications focused case study.

**Keywords:** Web Services, Service Orientated Architecture, Semantic Web, Web2.0, REST, Telecommunications.

## 1 Introduction

Telecommunication companies (telcos) are currently witnessing an erosion of their customer base due to increased competition and the emergence of 'Over The Top' (OTT) service providers – service and content providers that don't own the network they use – who threaten to disintermediate telcos to the role of a commodity provider (i.e. a supplier of 'dumb pipes'). In response to this threat, telcos are looking into ways to generate further income by offering their services, e.g. voice, messaging, etc. publicly on the web. Telcos have in this way the opportunity to reduce their costs, get products to market quicker and provide customers with the flexibility they are increasingly demanding. Furthermore the opportunity exists for telcos to transform themselves by adopting new business models e.g. based on service platforms or by utilizing the relationship they hold with their customers to create value.

A high proportion of these services are provided via RESTful services [1] (i.e. services conforming to Representational State Transfer) rather than classical web

services based on SOAP and WSDL. An internal BT survey found that 47% of providers supported REST whilst only 25% supported SOAP. The RESTful approach is generally simpler and aligns more closely with the web architecture. However, they are generally described in human readable web pages rather than machine readable XML files as is the case with WSDL. Thus enabling automated discovery and consumption of RESTful services is even more problematic than is the case with WSDL web services.

A further issue is the complexity of composing multiple services to meet a particular need (i.e. a mashup) or to integrate a service with an existing interface or system. As soon as a user has a need to go beyond what is offered by a single service (which they would typically interact with by filling in form fields and clicking a button), some development expertise is required.

SOA4All is an EU integrating project that supports the creation and proliferation of a "Service Web"—a Web where millions of parties are exposing and consuming millions of services seamlessly and transparently. A major outcome of the SOA4All project is SOA4All Studio, a set of online tools that cover the whole life-cycle of services from the end-user perspective: interaction with services is addressed from provisioning (where annotations are made on different types of services, and where they can be composed into more complex ones), consumption (where suitable services can be discovered and invoked) and analysis (where the execution of services can be monitored and analyzed at different levels).

By lowering the entrance barrier to the service world, SOA4All supports the "service prosumer", i.e., end-users who not only interact with services in a passive manner, consuming them, but are also able to create new ones or compose existing ones, etc. Telcos can leverage this fact by offering a set of base services that expose their telecommunications capabilities with which users can create new applications that make use of them, implementing new niche personalized services more easily.

This paper explores the SOA4All approach by describing a telecommunications-related scenario based on RESTful services and HTTP-based web APIs which is then implemented using SOA4All Studio through a process including description, discovery, composition, deployment and usage. In Section 2 we briefly describe the BT Ribbit API and identify the need for the SOA4All approach. The scenario and associated services are described in Section 3. The implementation of the scenario is described in Section 4 followed by an account of the next steps for the project in Section 5.

## 2 The Ribbit Vision

Ribbit (http://www.ribbit.com), a wholly-owned US-based BT subsidiary and a voice-oriented web company ('webco'), currently allows developers to consume voice services accessible via Adobe's Flex and Flash with a REST API currently in beta. The services that are exposed at the moment include voice calls, call routing, call management, third party call control, voice and text messaging, speech-to-text, VoIP and contact management, with more to come in the near future. Currently, users require detailed technical knowledge of the Flex, Flash or PHP programming

languages to be able to access, combine and use Ribbit's web services. In the SOA4All project, the use of contextual knowledge will support both the composition and provisioning of services in a customized manner. Using automation, we plan to shield service users from the complexities of creating such knowledge. We will also take advantage of semantic descriptions of services in building (semi-) automated provisioning, composition, consumption and monitoring tools. The next-generation platform we envisage will also enable inclusion of third party services. In addition, it will address several key issues for BT's transformation, including: (i) reducing time to market; (ii) enabling third-party services to be integrated into BT's portfolio; (iii) increasing so-called 'new wave' revenues from networked IT services; and (iv) extending BT's SOA to the public web.

The possibility of increased competition from the 'Over The Top' (OTT) service providers – service and content providers that don't own the network they use – highlights a risk that telcos could become 'disintermediated' from the digital supply chain. Webcos typically use advertising based business models, whereas telcos collect revenues through usage-based billing. As these two sectors converge, the challenge for telcos is to reconcile the two different business models, finding ways to generate revenue from advertising, while continuing to offer billable services where appropriate. Finally, other changes are apparent such as the rise in virtual social networking, the roll-out of alternative access networks such as WiMax and the emergence of enterprise mashups that use Web 2.0-style applications alongside more traditional IT assets.

Considering all these aspects, by appropriately positioning themselves in the Web 2.0 world, telcos will continue to evolve and transform themselves to providers of 'smart' pipes (connections backed by QoS guarantees and service level agreements), platforms that support an open service ecosystem and a range of telco and third-party applications that run on such platforms. Telcos will not only need to create new services to address the needs of the long tail (i.e. niche markets), but also allow third-party service providers to make use of telcos' underutilized operations and business support systems capabilities to create new service offerings, thereby creating new revenue streams.

## 3 Scenario Description

Our scenario describes a situation in which SOA4All technology is used for creating simple mash-ups of Ribbit services and other popular services on the Web. The aim is to make it easy for novice users to access Ribbit services and combine them with other services on the Web to create novel applications.

The focus of our scenario is on casual users building non-critical applications, and therefore involves minimal security or management infrastructure. In the scenario, a composition is created that allows one to organize a social meeting with a group of friends at short notice (as shown in Fig. 1).

The process is as follows: (i) Get list of friends from social networking site – in this example Last.fm was used, as it has a simple and open API to retrieve connections between users, and one of the options for meeting up is related to

concerts, which are also available in Last.fm (ii) Find out which of the identified friends are currently close to the user's own current location using mapping sites such as FireEagle and Multimap (iii) Filter the meeting location list depending on reports from a weather service (iv) Find out travel information for the proposed meeting

Send out invites and directions using Ribbit SMS to those users that are located within a defined range, customized depending on the weather information.

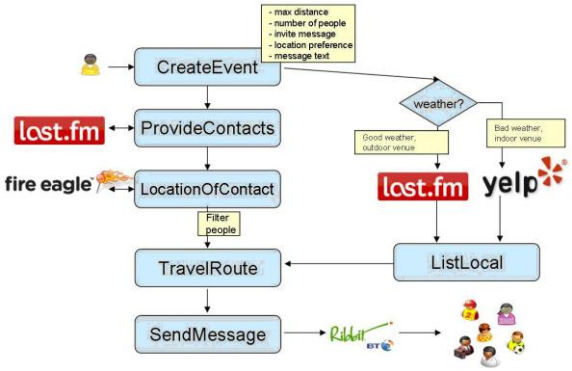


**Fig. 1**: Identifying service properties.

For this scenario, several existing RESTful services and HTTP-based APIs were used. These use a range of authentication methods such as OUATH (http://oauth.net/). or API keys and return data as XML or JSON (JavaScript Object Notation).

## 4 Composition of Restful Services Using Soa4all Studio

In this section we briefly describe the SOA4All project and the SOA4All Studio and then illustrate the Studio's use in the implementation of the scenario.

The SOA4All project will help to realize a world where a massive number of parties expose and consume services via advanced Web technology. The outcome of the project will be a framework and software infrastructure that aims at integrating SOA and four complementary and revolutionary technical advances (the Web, context-aware technologies, Web 2.0 and Semantic Web) into a coherent and domain independent worldwide service delivery platform.

The aim of the SOA4All Studio is to provide an integrating interface for the various components developed in the SOA4All project, namely, service provisioning, consumption, process modelling, and analysis. It provides two levels of service i.e. infrastructure services such as storage, communication and user management and a UI Library containing widgets, templates and a dashboard. The latter provides the connection between users and the SOA4All run-time components. Further details about the studio are provided in [2]

### 4.1 Service Description

The services used within our scenario are RESTful services with API descriptions in HTML pages. In order that these services can be more easily discovered and used in a service composition, they need to be supplemented with semantic annotation of their properties. This additional semantic information supports the automation of discovery, composition and consumption tasks, which otherwise have to be performed completely manually. In addition, as opposed to WSDL services, there is no widely accepted structured language for describing RESTful services. As a consequence, in order to use RESTful services, developers are obliged to manually locate, retrieve, read and interpret heterogeneous documentation of RESTful services in HTML, and subsequently develop custom tailored software that is able to invoke and manipulate them. In SOA4All, a solution to these challenges is provided by the provisioning platform [2], which as the name suggests, provides tools and functionalities for supporting the provisioning of services and in particular, semantic Web services. The creation of semantic RESTful services is enabled through SWEET – the Semantic Web sErvice Editing Tool [3], which is part of the Provisioning Platform Prototype. It enables both the creation of machine-readable RESTful service descriptions and the addition of semantic annotations, in order to better support discovering services, creating mashups, and invoking them.

Therefore, the first step of the scenario is to use SWEET to create semantic descriptions of the RESTful services. Each of the service descriptions is annotated following these four main steps:

**Identifying service properties** - Insertion of hRESTS microformat tags in the HTML service descriptions in order to mark service properties (operations, address, HTTP method, input, output and labels). This is achieved by highlighting the relevant portion of the description and clicking on the appropriate hRESTS tag in the tool and is shown in Fig. 2.

**Identifying suitable domain ontologies** - Integrated ontology search for linking semantic information to service properties. The right of Fig. 2 shows a domain ontology being used to annotate a service property (password).

**Semantic annotation of service properties** - Insertion of MicroWSMO [4] model reference tags, pointing to the associated semantic meaning of the service properties.

**Saving the semantic RESTful services** - Saving semantically annotated HTML RESTful service descriptions and automatic extraction of RDF MicroWSMO service descriptions based on the annotated HTML into the SOA4All repository.

In accordance with good practice, rather than create new ontologies, we re-used publically available widely used ontologies such as the W3C's GEO vocabulary for latitude and longitude, the FOAF vocabulary for personal profiles and the Kanzaki music vocabulary for music events information. An additional ontology was engineered for any missing properties.

When the user has finished annotating the HTML RESTful service description with hRESTS and MicroWSMO tags, the resulting annotated service can be saved, exported to RDF and / or uploaded to a service repository allowing service consumers to locate it.

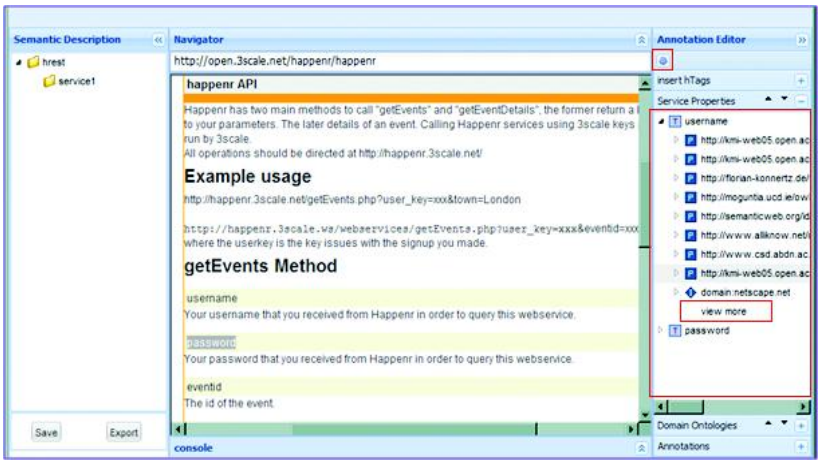


**Fig. 2**: Annotation of a service within the SWEET tool

### 4.2 Service Discovery

Service-orientation advocates the development of complex distributed applications based on the reuse and composition of existing functionality offered as services. Essential to this vision are the publishing and discovery of services. A number of repositories for WSDL services have been created to this end, e.g., UDDI [5], but they have failed to provide suitable support for publishing and querying them in an expressive and extensible manner. Research on SWS has devoted significant effort to enhancing service repositories with semantics in order to provide more accurate results or simply to automate the discovery of services to a greater extent, see for example [6] and [7]. Despite the efforts thus far, the largest public SWS repository is probably still OPOSSum, a test collection with less than 3000 service annotations [8]. Unfortunately, although useful for testing, OPOSSum does not represent nor does it aim to be a service repository supporting the publication and discovery of services.

In order to achieve the vision previously highlighted it is essential to have in place appropriate mechanisms for supporting the crawling, publication and querying services available on the Web, may they be WSDL-based on Web APIs.

To this end we have developed iServe [9], a platform for the seamless publishing of SWS. iServe addresses the publication of services from a novel perspective based upon 3 fundamental lessons learnt from the evolution of the Web of Data [10]: (i) lightweight ontologies together with the possibility to provide custom extensions prevail against more complex models, (ii) linked data principles are an appropriate means for publishing large amounts of semantic data, both for human and machine consumption and (iii) links between publicly available datasets are essential for the scalability and the value of the data exposed.

iServe (http://iserve.kmi.open.ac.uk) provides support for the seamless publishing of SWS defined in a variety of formats and according to different conceptual models, by transforming them into linked data that can easily be browsed, retrieved and

querying by both humans and machines. The current version of iServe provides support for SAWSDL [11], WSMO-Lite [12], MicroWSMO [13], and (partially) OWL-S [14] descriptions. Taking the original descriptions, iServe automatically generates the appropriate RDF statements according to a common and minimal service model largely based on the one defined in WSMO-Lite, and exposes them as linked data in a manner that is suitable for the description and interlinking of services, people and data.

To facilitate the consumption and manipulation of the published linked data about services, iServe provides three interfaces: (i) a Web-based application, called iServe Browser, allowing users to browse, query and upload services to iServe, (ii) a SPARQL (the W3Cs RDF query language) endpoint where all the data hosted in iServe can be accessed and queried and (iii) a RESTful API that enables creating, retrieving and querying for services directly from applications.

Additionally, to further simplify the publication of services we provide two Web-based applications, SWEET (mentioned earlier) & SOWER supporting the annotation of Web APIs and WSDLs based on MicroWSMO and WSMO-Lite respectively, which directly provide the means for publishing the annotations in iServe

### 4.3 Service Composition

SOA4All Studio contains a Process Editor tool which is used to link the different (annotated) services into the desired process (composition), also taking into account the restrictions that apply in the scenario. It is underpinned by the semantically powerful LPML [15] (Light-weight Process Modeling Language), yet the graphical interface users see is simplified for ease-of-use. The missing information is either supplied by users where they understand the underlying semantics, or by a reasoning engine which considers data types and activity specifications.

In this section, we first overview the underlying composition operators required to support the end-user in modeling her composition, Then, we briefly explain how expert and novice users interact with the process editor to model their composition using some specific composition operators.

#### 4.3.1 Service Composition Constructs

Any process is composed of mashing data (i.e., data-flow based operators) from one service to another. For instance, the following illustrates the data-flow based operators used to model and design the composition of our scenario: (i) splitting data from the end user inputs e.g., “Location User” is reused in different points of the composition; (ii) merging data from different services’ outputs e.g., the getLocalVenue and getLocalBar services provide multiple outputs that require merging for processing in the next step (by some following services); (iii) filtering data from the output of a service e.g., the SendSMS service will need only the first (closest) Location venue provided by the getLocalVenue service and getLocalBar service outputs; (iv) iterating processes on specific data controlled by some rules e.g., each element of the getFriend service needs to be processed by the getLocation service as inputs. The rule used is “Count Operator” with the “number of people” as variables; and (v) counting the

number of outputs provided by a service e.g., the getLocation service iterates on a given number of outputs of a service, here a given number of friends.

In addition, the latter process is composed of one specific control flow construct supported by our ongoing work on LPML i.e. Conditional branching on services depending on the values of output data of services e.g., the getWeather service moves to getLocalVenue or getLocalBar depending on the data provided by getWeather service.

To this end, the scenario has been modelled and encoded using the LPML composition language. More specifically, the whole composition has been specified using a subset of the "Data Flow" operators (enabling modelling of a composition as a Mash-up of data) provided by the defined language: (i) Split Operator: This operator receives an input and splits it into two or more identical outputs. This operator can be used when the end-user wants to perform different operations on data from the same input (ii) Merge Operator: This operator takes an arbitrary number of inputs and produces an output, which is composed of the merge of its inputs. (iii) Filter Operator: This operator can be used to include or exclude items from an output of a service. Therefore, some rules can be created on top of the LPML language to compare the output of services to values that the end-user specifies. (iv) Loop Operator: This operator introduces the idea of sub-data processing. Any other operators could be inserted inside the Loop operator. An output of a service is provided to the Loop operator, the sub-data processing is run once for each item in the latter output. (v) Count Operator: This operator counts the number of items in the input and outputs that number. In addition, a subset of the "Control Flow" operators provided by the defined language i.e. If_Then_Else Operator: This operator enables branching from one service to another depending on some logical conditions.

#### 4.3.2 Service Construction for Users

All the data flow operators above are supported by the process editor (Fig. 3) of our framework but they are not explicitly represented at the top representational level. Apart from a simple representation of boxes (activities) and arrows linking them (flow), their precise meaning is inferred or solicited at a later stage, depending on the skills level of the current user, in a process illustrated below.

The end-user is in charge of modelling her composition by dragging and dropping activities to the workspace page (see Start Activity in green, Activities in Orange and End activity in red – Fig. 3). Then, for each activity, the end user binds the service that fits her goals for this activity.

Once the activity and service selection is achieved, a chaining phase is required to connect services together and satisfy the data flow of the overall composition. To this end, two modes are possible, depending on the background of the end user.

In the case of an expert user, it is up to her to select the category of the data flow connection (Merge, Filter, Split …) she expects between the services. For instance, between the getLocation (from FireEagle) and getDistance (from Multimap) operations, the user can filter the data coming from the getLocation operation, or merge data from this service and others to the getDistance operation. To this end, the end-user has access to data descriptions of services (at semantic and syntactic levels, depending on the services descriptions) and then could draw connections between services parameters sharing some similar or close descriptions. Then, background

tools operate the final data connection between services by means of some Connectors (not described in this paper).

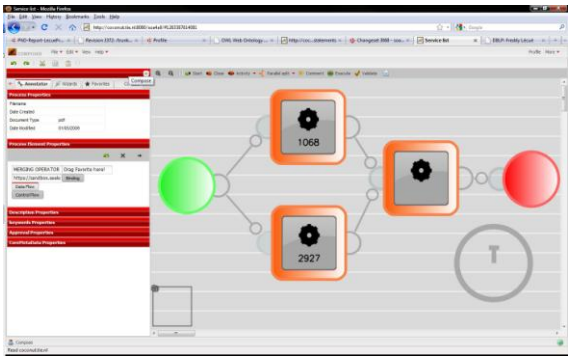


**Figure 3**: Process Editor.

Alternatively, in the case of non expert users, the process editor interacts directly with the reasoning engine (that comes with the SOA4All platform) to automatically infer the most appropriate connection between appropriate parameters of services, depending on the semantic descriptions the services expose. To this end, the process editor requires that connections between activities (and their services) are simply designed by the end-user. Obviously, the inferred connections can be subject to change if the user expects different data manipulation in the designed composition.

After the composition, its services binding, the control and the data flow designed on the workspace of the process editor, the composition can be saved (through a serialization in a service composition language a la BPEL4SWS [16], validated and finally executed by the execution engine of the SOA4All platform.

### 4.4 Service Execution

In SOA4All the execution of the service is handled by SPICES (Semantic Platform for the Interaction and Consumption of Enriched Services)[17]. SPICES will enable the composed service described in our scenario to be executed since it supports both WSDL and RESTful services (including combinations of the two).

The contribution of SPICES is in both supporting the end-users interaction with the service(s) and in supporting the consumption process itself which includes lifting and lowering between the conceptual level (which is tailored to humans) and the execution level (which is the actual service API).

The user is supported via the generation of appropriate user interfaces for the service. Typically, the user is presented with a set of fields which must be completed to allow the service to execute. Since the context of each field may not be obvious from a textual description attached to each field additional support is provided which allows the user to visualize the RDF graph appropriate for each input. This makes use of the ontological annotations referred to earlier which may include comments and

relationships to other concepts and supports the user in understanding the context of the required input.

SPICES also provides support for the personalization of user interfaces. This makes use of individual user profiles to ensure that interfaces are built in accordance with preferences or contextual information e.g. language, currency, current location and to ensure information that is already in the profile is automatically entered so that the user is not asked to provide it again. This is supported via subsumption based reasoning to semantically compare the service descriptions and the profile and context descriptions associated with the user. Thus in the scenario, one required input is the mobile phone number of the user. Since this is contained in the user's profile, it can be automatically provided by SPICES.

The final area of user interaction support is that for authentication. In the scenario, a variety of authentication types are required to invoke a service including API keys (e.g. for Last.FM) which must be applied for and more general methods such as OAuth (e.g. for FireEagle). When the services are annotated the form of authentication is included and as a result SPICES is able to provide the required details from the user profile or support the user in obtaining the required details if the profile does not contain them.

From the service execution perspective, SPICES builds upon previous work on lifting and lowering [18] which supported WSDL services by moving from RDF instances to XML messages (lowering) and back again (lifting) to include support for RESTful services where XML is not required or applicable. With REST, an HTTP request needs to be built. To do this, SPICES uses UriTemplate [19] which allows HTTP requests to be built using a template with placeholders for variables which can be filled in with the appropriate values at run-time. The UriTemplate is stored with the service annotation. For example, a call to the Ribbit SMS service would be built from the following template:

```
http://ngwr.labs.bt.com/Ribbit/myapp/SendSMS.php?recipi
ent=tel:{mobile}&message={messageText}
```

where {mobile} and {messageText} are variable names which are populated by input or profile values. Further work is underway to cope with POST messages where the message body as well as request header must be modified.

Regarding lifting, services typically respond with XML or in some cases JSON. As such the lifting approach for WSDL services is applicable i.e. XSLT-based schema mapping (with appropriate prior conversion to XML for JSON-based responses).

SPICES includes service execution components for both WSDL and RESTful services. After lowering has taken place, the correct invoker is chosen (again using the service annotation) and called to interact with the target service. The response can then be passed on the lifting component.

## 5 Next Steps

In section 4 we described the various steps in the process of enabling mashups based on RESTful services using a telco related scenario. We have developed a prototype

using SOA4All Studio which incorporates service description, discovery and composition. Service execution is under development and is our short-term focus.

In addition to this there are a number of other areas upon which we intend to focus. We are developing an additional scenario that fully reflects the business models described in section 2 i.e. where businesses (rather than casual users) are composing and consuming services from a variety of providers and offering these to their end customers with an appropriate 'service wrap' including billing, authentication, etc.

The first area we will address concerns the monitoring of services in order to be able to track certain non-functional properties such as the response time or the availability of services in order to provide a more adaptive environment. Both periodic batch as well as runtime monitoring will be carried out in order to gather information about services. This information shall be used for ranking when users are searching for services to use but also to support service providers to track the service delivery and take corrective measures if necessary. Rule engines shall be used to support the latter.

Finally, we will promote the creation of a community around services. This community will on the one hand create and share new composite services that will gradually enrich the available functionality providing added value solutions on an increasing complexity. On the other hand, members of the community will provide highly valuable information about the services themselves, may it be directly through ratings and comments, or indirectly through invocation. On the basis of this information the platform will support more accurate rankings of services and will include support for recommending services to users based on their profile.

Both of these latter areas will also be explored in our business reseller scenario.

## 6 Conclusions

Telecommunications companies are turning to web-based APIs as a way to allow direct access to their capabilities, many of which adopt a RESTful approach in order to create new revenue streams. In order to promote ease of use and enable mass market adoption the complexity involved with describing, finding, composing and using these services must be drastically reduced with a greater reliance on automation.

We have described the approach of the SOA4All project in providing support for RESTful services which until now have been largely neglected by efforts to deliver 'semantic web services'. The application of this approach to the Telecommunication domain is expected to enable a greater level of adoption of such services by both expert and novice users for both personal consumption and for the creation of new business opportunities.

## REFERENCES

1. Pautasso, C., Zimmermann, O., Leymann, F.: RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision, 17th International World Wide Web Conference (WWW2008), Beijing, China (2008)

2. Domingue, J., Fensel, D. and González-Cabero, R.: SOA4All, Enabling the SOA Revolution on a World Wide Scale. Proceedings of the 2nd IEEE International Conference on Semantic Computing ICSCIEEE Computer Society , August (2008)
3. Maleshkova, M., Pedrinaci, C., and Domingue, J.: Supporting the Creation of Semantic RESTful Service Descriptions Workshop: Service Matchmaking and Resource Retrieval in the Semantic Web (SMR2) at 8th International Semantic Web Conference, October (2009)
4. Vitvar, T., Kopecky, J., Viskova, J., Mocan, A., Kerrigan, M. and Fensel, D:. Semantic Web Services with lightweight descriptions of Services, Advances in Computers, Elsevier, Volume 76. (2009)
5. Clement, L., Hately, A., von Riegen, C. and Rogers, T.: UDDI Specification Version 3.0.2. Technical report, OASIS.(2004)
6. Srinivasan, N. Paolucci, M. And Sycara, K:. Adding OWL-S to UDDI: Implementation and throughput. In Proceedings of 1st International Conference on Semantic Web Services and Web Process Composition (2004).
7. Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S. and Miller, J:. METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. International Journal of Information Technologies and Management, 6(1):17–39. (2005)
8. Küster, U. and König-Ries, B.: Towards Standard Test Collections for the Empirical Evaluation of Semantic Web Service Approaches. International Journal of Semantic Computing 2(3), December (2008).
9. Pedrinaci, C., Domingue, J., and Reto Krummenacher: Services and the Web of Data: An Unexploited Symbiosis, Linked AI: AAAI Spring Symposium "Linked Data Meets Artificial Intelligence", Standford, USA, March (2010)
10. Bizer, C., Heath, T. and Berners-Lee, T. Linked data - the story so far. International Journal on Semantic Web and Information Systems (IJSWIS) (2009)
11. Farrell, J. and Lausen, H:. Semantic Annotations for WSDL and XML Schema. http://www.w3.org/TR/sawsdl/, January 2007. W3C Candidate Recommendation 26 January (2007).
12. Vitvar, T., Kopecky, J., Viskova, J. and Fensel, D.:. Wsmo-lite Annotations for Web Services. In Hauswirth, M., Koubarakis, M. and Bechhofer, S., editors, Proceedings of the 5th European Semantic Web Conference, LNCS, Heidelberg, June (2008).
13. Maleshkova, M., Kopecky, J. and Pedrinaci, C. 2009. Adapting SAWSDL for Semantic Annotations of Restful Services. In Workshop: Beyond SAWSDL at OnTheMove Federated Conferences & Workshops.
14. Martin, D., Burstein, M., Lassila, O., McDermott, D., McIlraith, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N. and Sycara, K.: OWL-S: Semantic Markup for Web Services. http://www.daml.org/services/owl-s/1.0/owl-s.pdf, (2004)
15. Schnabel, F., Xu, L., Gorronogoitia, Y., Radzimski, M., Lecue, F., Ripa, G.: Advanced Specification Of Lightweight, Context-aware Process, SOA4All Deliverable D6.3.2,http://www.soa4all.eu/file-upload.html?func=fileinfo&id=127
16. Nitzsche, J., Norton, B.: Ontology-based Data Mediation in BPEL. In: BPM Workshops. 523–534 (2008)
17. Álvaro, G., Martínez, I., Gómez, J., Lecue, F., Pedrinaci, C., Villa, M. and di Matteo, G.: Using SPICES for a Better Service Consumption, Poster at Extended Semantic Web Conference (2010)
18. Kopecky, J., Roman, D., Moran, M. and Fensel, D.: Semantic Web Services Grounding. In Proc. of the International Conference on Internet and Web Applications (2006)
19. Gregorio, J., Hadley, M. and Orchard, D.: URI Template. IETF Draft. http://tools.ietf.org/html/draft-gregorio-uritemplate-0 (2008)