Smart Data Web Services

Cătălin STRÎMBEI
Al. I. Cuza University of Iasi, Faculty of Economics and Business Administration,
Department of Business Information Systems
linus@uaic.ro, catalin.strimbei@gmail.com

In the new world where the Internet business resembles with a large and distributed sea of links, using Cloud architectural model, the web-service interoperability and SOA model one could deploy an arguably new class/generation of apps/services that could leverage the marriage of these originally distinct computing models to be real smart, as autonomous, dynamic and agile, but open to integrate and adapt.
Keywords: *Web Services, Cloud Computing, SEO, SQL, XML*

1 Introduction

The *Web Service Technology* seems to have passed the over-hype phase by trying to reach to the maturity level that will finally enable its mass adoption. The Cloud computing era could be the trigger factor to overspread the web services as the foundation or the platform of choice for the Internet applications or, at a larger scale, for the Internet business systems.

In this regard, we believe that in the context of the marriage with cloud computing paradigm, the web service architecture could achieve some special advantages in the following directions:

- data access openness and standardization;
- autonomy of underlying supporting infrastructure;
- dynamic search and discovery using the already widely spread searching technologies like SEO (as Search Engine Optimization).

In fact, we believe that a new web services generation could come (must come?) on the stage of the business applications and technologies, in order to leverage the true opportunity of Cloud architecture in the Internet-based business processes area. The Cloud services providers made some strides on storage and office-based applications, but for the actual business platforms and application services the big wave is yet to come over.

In our opinion an enhanced support of business processes and business functions will have a major impact on the proliferation of the Cloud-based architectures. The current SaaS model (Software as a Service) that delivers business oriented applications has some serious limitations that slow down its adoption:

- preserving a quasi-monolithic approach regarding aggregation of the available business functions (integrated, but not enough modularized);
- deep dependency on the backend Cloud infrastructure (limited autonomy and agility);
- poor integration with other business oriented services, as they aren't built with large-scale inter-connectivity in mind, and preserving a quite inflexible layered architecture that is clearly delimited between the boundaries of a business system template (limited openness to interchange data protocols).

As we will argue in the followings, the "smartness" capacity of the new kind of web services will be sustained on a sum of characteristics coming from *three service models* (see Fig.1), characteristics as *agility, openness, dynamic, autonomous*.

These characteristics could leverage a higher architectural level consisting in smart data integration in the web context, meaning: no more proprietary drivers to access data sources, no more proprietary and private encrypted data formats, no more static referencing or integrating data sources (smart as dynamic discovery and reference of valuable data sources).

We think that the business oriented web services will have to deal with, among others, at least two fundamental challenges, as they are stated in the SOA blueprints:

- *business processes control or orchestration*; one potential initiative in this area could be a form of standardization of some kind of event-based web services (we intend to argue and develop such architecture in a future paper);
- *data query, data interchange and data management*; the current paper will try to argue and outline a possible feasible architecture.

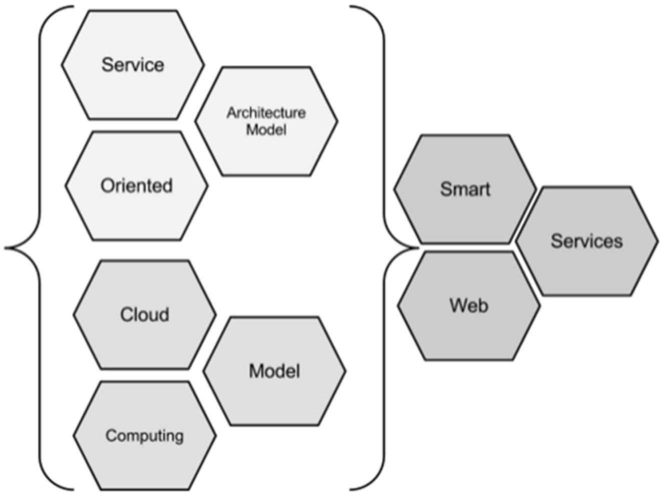

Fig. 1. A Vision on Cloud Computing and Web Services marriage

Concerning the special kind of *data centric web services*, we think that they have some characteristics that could make them even better suited with/into cloud based architectures, because:

- they are *structured* (borrowing some features from databases interoperability with traditional data oriented software components of the business systems) thus assuming:
  - structured (or, at least, semi-structured) data definitions (metadata);
  - structured and declarative data requests;
- they are *data intensive*:
  - taking into account the large data amount to interchange;
  - taking into account the data integration issues (the need for interchangeable data format);
  - taking into account the large data amount for storage;
  - taking into account the data processing computing capabilities (such as OLAP or Data Mining);
- they could be integrated into larger/aggregate/composed/derived structural architectures, thus implying:
  - data consolidation of heterogeneous data systems (integrated OLTP systems);
  - data aggregation for analytical processing specific to decision-making business processes (such as OLAP-based systems).

In this paper we will try to argue the advantages and the appropriate engineering principles to support such cloud-based data centric web services.

2 Backgrounds

As current state of the field we will present three service models which we consider the most influential to the technological features of the current Internet business systems.

SOA and Web Services Models

The Service Oriented Architecture propose a new and revolutionary distributed component model, mentioned sometimes as service oriented computing, and aimed to increase the efficiency and productivity [1:25-66] of business processes supported by the emerging Internet-based application systems.

SOA has the merit to introduce a new kind of technological "democracy" where the application systems are considered a federation per se, thus opening the doors to a new kind of logical distributed computing approach where the technological platforms are downgraded to the implementation or physical level. Here we have to mention that database design methodologies and systems have introduced/recognized for quite some time a system with three architectural levels: conceptual, logical and physical [2], [3]. In this context, the business process design with languages like BPMN covers the conceptual level (or the organizational, enterprise level), with languages like BPEL and web services specification covers the logical level, and with web services implementation platforms (like JEE or .NET) covers the third level, as one can see in Figure 2.

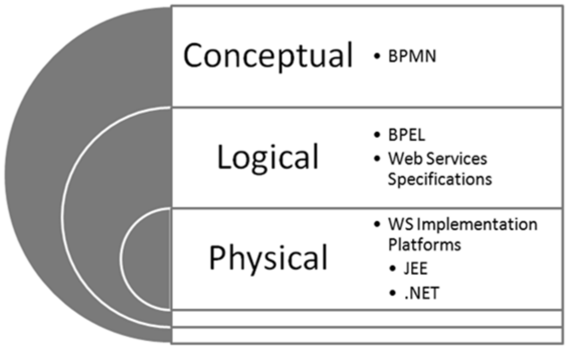

Fig. 2. SOA architectural levels

In other words, the *data independence* principle has to be recognized and could be applied also in the SOA context, because we think that this kind of architectures is indeed compatible with it. Although the database systems have recognized this principle and have even standardized the data access level, the interoperability issues still remained and broke the independence principle, consequently the federated database systems became a very complex and difficult research topic because of the heterogeneity of database systems and of the heterogeneity of their access protocols.
SOA has been conceived as an evolutionary stage in the technological landscape of the distributed computing, but, although it borrowed many principles and design patterns [20] from the traditional distributed solutions and methodologies, we think that, in the end, this effort proves to be disruptive and revolutionary if we take into account, beyond the technological consequences, the (not so side) effects on the Internet business: e.g. virtual organizations become a practical kind of business enterprise in the specific context of SOA (see [4] and [5] for a methodological argumentation of virtual organizations or enterprises).
The need of *service* orientation [1: 81-85] came from the effort to surpass an entire class of complex issues and traditional solutions that, ultimately, ended by mixing the logic of business rules, the component integration and the physical interoperability into the same context or layer. The *service* principles stated to overcome these problems are the following:

- service loose coupling to minimize dependencies;
- service abstraction to minimize the availability of meta information;
- service composability to maximize the interoperable possibilities;

and also:

- *standard* service contract;
- service *reusability* to implement generic and reusable logic contract;
- service *autonomy* to implement the functional boundaries independently from runtime environments;
- *statelessness* services to free service logic from their state-management;
- service *discoverability* to implement communicative meta-information.

Consequently, one of the primary merits of the *services* consists in their characteristic of being agnostic to the application-specific logic with some important direct or indirect consequences, as those presented in [1:81-84]:

- increasing of data *consistency* and behavioral consistency and *predictability*;
- decreasing the dependencies in the same time with increasing *reusability* regarding business processes and configurations (increase agility).

SOA presumably involves some kind of centralized orchestration model as the SOA design patterns require; in this regard one could review the Service Controller pattern and the Workflow Connector pattern presented in [6]. The SOA architectural style imply that the involved services become, more or less, parts of an aggregate structure; therefore having a limited area of responsibility and action, as one could conclude from the discussion on service autonomy presented in [1:293-323]. Thereby the SOA "canonical" *service model*, with its "classic" service categories and their formal possibilities to integrate (couple), to reuse and to compose, proposes a loose-coupling architecture but in the context of a centralized approach, as in Fig.3. We think that this kind of model could limit, at some degree, the *agility* objective of SOA, and an alternative model centered on dynamic *discovery* and *composability* (*autonomous* services model) could leverage the context of Web services.

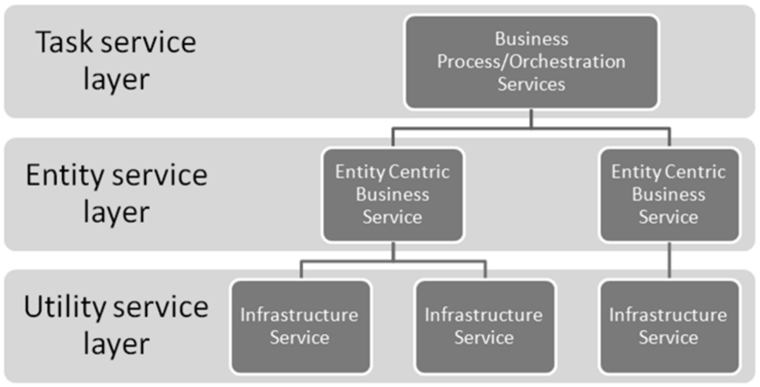

Fig. 3. "Canonical" SOA Service Model

Web Services as a logical foundation of SOA
In our vision, the *services* are conceptually grounded as being "beyond the objects" at the architectural level (the Web Service Model is outlined in Fig.4). One of their most "suggestive" principles comes from the object oriented *interface pattern* [7], but, additional, the web services are agnostic to the programming level. In fact, the architectural principle of "separation of concerns" is generically assumed by the specific nature of the *service* concept. The services from the SOA architectures are somehow distinct from the *Web Services* due to the "technological agnostic" SOA principle [1:114-115]. Consequently, the enterprises have the freedom to choose the solution to declare or to specify the formal service definitions and the freedom to choose the implementation technology of the actual service components, as proprietary (EJB, .NET business components), message-oriented or Web based frameworks. In this context, the *Web Service Technology* has developed as the most appropriate "logical" foundation of SOA [1:45-50], to the extent that it adheres most "naturally" to the SOA service principles due to its extensive standardization (WSDL, XSD, SOAP, UDDI and WS-* extensions), contract-based inter-operability and document-oriented data interchange protocols. In fact, many consider that WS-Architecture as being a distinct and more "liberal" architectural style of SOA.

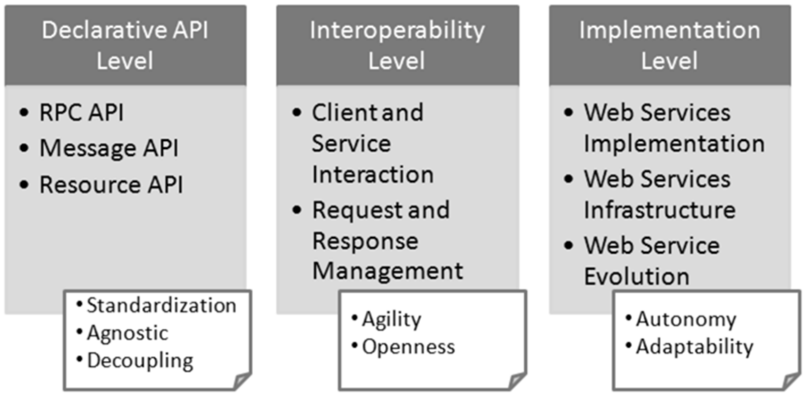

Fig. 4. Web Services Model

Cloud Platforms for Web Services
The Internet ready applications and the web hosting had a determinant influence to the "dot com" or "eCommerce" revolution. Beyond the broken economic model of the first generation of these business technologies and architectures, the cloud-based initiatives seem to be an evolutionary step ahead, from both points of view of business model and of technological model. In this context, it is important to mention that the companies that survived and proliferated despite the "dot com bubble burst", like Google or Amazon, have reconsidered and have redesigned their web based applications and platforms from the ground up. In their vision, an iBusiness (as an internet or web-based business) or an iApplication (as an in-

ternet or web-based application) will be a *link* (an URL or URI) in a world of inter-related *links* [8:24-29], in fact a distributed model pulled up to an extreme. In this context, the key elements of cloud computing model [9:4-5] are:

- computing resources, wrapped up as commodities for web access;
- extremely easy access to web resources for clients (end users);
- business/economic model based on "pay just for what you use" principle.

In other words, cloud computing means *dynamically delivering of scalable, elastic, shared and virtualized resource* as *services* accessible over the *Web* [10:2-4].

Generally, the cloud services are considered to come in three flavors [10:16-17]:

- **SaaS**, Software as a Service, like Salesforce.com services, is considered as an evolution of the earliest Cloud initiatives named ASPs (Application Server Providers) that focused on delivering highly customizable and end-user oriented packaged and deployed applications that run entirely on the vertical infrastructure of the provider. One of the main evolutionary characteristics of SaaS is the customizability that it is heavily based on metadata interpreted at runtime and being, obviously, user-oriented (or client-oriented).
- **IaaS**, Infrastructure as a Service, with Amazon EC2 as the flagship initiative, is a questionable but determinant step to deliver more control of cloud resources to the clients. If SaaS could be considered as very business oriented and on top level of potential Cloud services, IaaS is considered as operating on the other end and is the area were virtualization of computing resources (software and hardware) gain full traction. In this context, one could build SOA architecture with full responsibility on managing and optimizing the supporting resources for services, having to take care to all dimensions of the architectural pyramid, but with a highly effective cost control.
- **PaaS**, Platform as a Service, with Google as one of the leading providers, is a veritable disruptive paradigm and has a major impact on the Web Services programming models: if IaaS allows to virtually install, manage and deploy software components on a full-blown application server in a very similar way as in a traditional, private context, PaaS should provide the control of the application server specific resources (but not to the application server itself) yet the performance levels will be controlled as a quality agreement with the provider. Consequently, Google App Engine could be viewed as a global, unique but highly distributed Application Server Environment. In this context the cost control could be even more effective being even more granular as in IaaS case: per open sessions, per user numbers, per calls etc. PaaS offers a very unique advantage for web service development and deployment: the clients should concentrate only on the business logic and on the architectural business aspects and should be less concerned on the low level performances or on other physical aspects.

No matter in what format cloud computing is delivered, the *abstraction* and *virtualization* [11:91-198] are the basic features of the Cloud model, presented in Fig.5, features that makes Cloud computing extremely complementary to Web Services, so that very often one can use just the simple term of *Cloud Services*.

We think that the IaaS strategy has an advantage over PaaS by potentially preserving the in-house services architectures that will be more likely to migrate to this kind of cloud computing, as the client could replicate their original deployment environment by configuring accordingly the cloud context. On the other side, the PaaS approach forces to redesign the deployment procedure and even to redesign the application logic of the existing web services, this way favoring building a new generation of web services to support business processes, but, in the same time, taking advantage of a new *service programming model*, as MapReduce [9:131-143] or Dev2.0 [9:143-157].

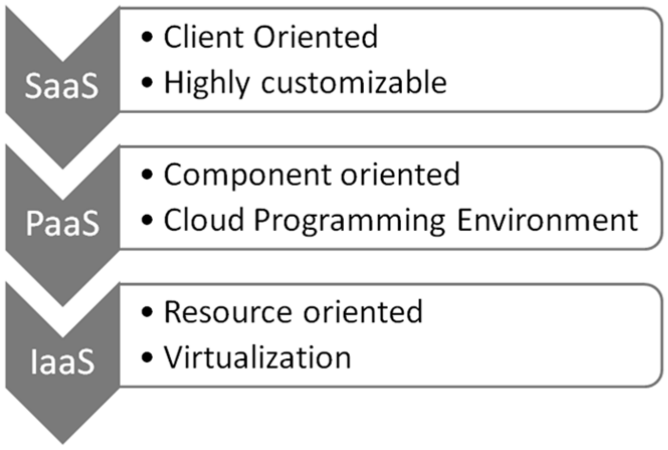

Fig. 5. Cloud Service Model

As the IaaS architecture resembles with the traditional but virtualized physical infrastructure of resources going down to the operating systems, the PaaS architecture abstracts the service resources like application engines, data stores and even file systems, along with some very active resources as memory cache or schedulers. Our opinion is that the PaaS architecture seems to be more compatible with the "canonical" service model from Fig.3.

The Cloud architectural platforms and their economic models focus mainly on performance (e.g. scalability) and costs. We think that this model is "criticisable" from some specific points of view:

- the cloud platforms tend to be proprietary and tend to limit the service deployment possibilities and make difficult if not impossible the deployment switching;
- even with many providers, the market seems to be excessively consolidated thus the clients easily become captive and this fact could prove a broken business model that is not focused on client, but on platform.

Consequently, even with the flexibility of the IaaS or PaaS model, the portability issues keep being relevant. We think that the standardization of each of the already established Cloud models (like Open Cloud Computing Interface, OCCI, family of specifications presented in [12]) has to be a desirable and that could prove to be a evolutionary step so that Cloud technological service model and Cloud service economic or market model could reach a real maturation level.

3 Discussions

In the background section we have tried to outline some of the defining characteristics of the paradigms of the *three service models* that will shape the Internet business of tomorrow. Our opinion is that these three service models have a set of compatible and complementary objectives and features. Consequently, we think that there is a need to converge them into a common but more powerful meta-model of services. The Web service model seems to be a natural fit for the SOA architectural model, and, in the same time, web services are becoming a determinant factor for the Cloud computing proliferation, especially in the PaaS format (see Table 1).

Table 1. Service Model Convergence

SOA Service Model	***Cloud Platforms***
Task service layer	WS as applications deployed on SaaS
Entity service layer	WS as data providers from DBaaS
Utility service layer	WS as web resources on IaaS
SOA as Platform as a Service (PaaS)	

The *Data Web Services* will be our use case that we will use in the followings to argue the *convergence* which could augment the Web Services determinant features. We will finally call them *Smart Web Services*.

In this context, the data centric services have some particular requirements that make them illustrative to analyze:

- (data) modeling requirements;
- (data structure) representation requirements;
- (data) resource (linking) requirements.

There isn't a *formal definition* of the Data Web Services, but there are several approaches, related to data implications on the web services, like:

- the use of web services for *accessing data and building distributed applications* [13];
- SSOA, as semantic SOA, that implies *ontology-based data exchange* [14];
- the web services whose behavior is determined by their *interactions with a repository of stored data* [15];
- the web service architecture to *optimize the exchange of large* (XML) *data volumes* [16];
- the collection of interrelated data web services *to handle enterprise data access* [17].

Presumably every kind of Web service uses some kind of data interchange procedure, at least to evaluate the parameters or arguments of web services operations. But there is a "class" of web services to which data represents the determinant or the dominant factor at least from the business logic perspective, relative to either:

- data interchange protocol (data access);
- data interoperability architecture;
- data resources (internal data storage).

From the SOA perspective the "most" data centric services revolve around the entity service layer [18:28], [1:485]. Some entity services will tend to have a coarse-grained functional scope because they are composed from other more granular data (or document) oriented along with their agnostic CRUD aggregated operations. For that matter, the *entity services* are sometimes presented as *entity-centric business services* or as simple as *business entity* services. They don't necessary orchestrate business processes but they could be the common denominator of a set of inter-related *process services*, *business process services*, or simply *orchestration services*. Even though their database support is not obvious (the entity services being technologically agnostic and their ultimate scope being to assume the conceptual role played by database in traditional business information systems) are very likely to encapsulate database resources or to invoke infrastructure services that access inner databases.

The database support or database implications on entity services are more relevant from the perspective of their technological implementations on Cloud computing platforms. In this regard, the newly emerged Cloud platforms and their providers embraced two kinds of approaches [9:117], [10:136]:

- to provide database infrastructure services through IaaS platforms, thus preserving the traditional database technologies enhanced with IaaS specific scalability features, e.g. Amazon EBS Relational Services, Google SQL Cloud, and Oracle12C expected database version;
- to provide a new data storage infrastructure, thus entirely changing the database paradigm and aiming even to replace it with a new kind of storage services and a new kind of data oriented programming model, like Google's DataStore based on BigTable data model with MapReduce programming model, Amazon's SimpleDB data model with Hadoop on EC2 programming model.

Consequently, relational database systems have survived in the new era of cloud-based data storage, but facing great competition from so-called NoSQL systems designed to be highly scalable especially for reading access, but, in our opinion owning a broken data integrity model.

In our opinion, there are three major Data Web Services flavors that form a variable geometry that could cover one or more of these three perspectives:

- *Business Centric* as Entity Services, whose primary role is structuring and orchestration of business entities;
- *Distributed Data Access Centric,* as Database Access and Integration Service, DAIS [19];
- *Data Storage Management Centric,* as heterogeneous and distributed data storage and scalable in memory (active) and persistent workloads.

We outlined that the service model of SOA could be completed with the Web Service Model, and, in the same time, could be completed with service model specific to Cloud Computing. In our opinion the intersection or the overlapping area of those three service models could define a new, enhanced or "smarter" service model.

In our vision, the *Smart Data Web Services* could be defined through an inter-related set of "smart"

features or capabilities like: dynamic and adaptive interoperability, autonomy and agility.
Analyzing this complementary approach, we have identified at least three directions where the cumulative effects of the three outlined service models could make "smarter" the *Web Data Access Services:*

- dynamic data interchange with dynamic metadata;
- service discovery enhanced by specific techniques like Search Engine Optimization;
- distributed linked service queries;
- dynamic resource (as storage) service acquisition (for autonomy and agility).

4 Experimental Projects

In order to build a feasible technological framework to productively develop Data Web Services, we are implementing some experimental projects taking into account those directions outlined above.

Dynamic data interchange with dynamic metadata
There are several initiatives related to the normalization of the data access and the data interchange in the Web Services context, like the WS-DAI standard specifications [19] or the SOA framework of the View-based Model-driven Data Access Architecture VMDA [21]. In this context, we have some practical achievements starting with the Service Data Objects specifications [22] whose defining features refer to:

- unifying the heterogeneous data resource access across different data source types;
- unified support for static and dynamic data types;
- support for disconnected programming models;
- enabling applications, tools and frameworks to more easily query, view, bind, update, and introspect data.

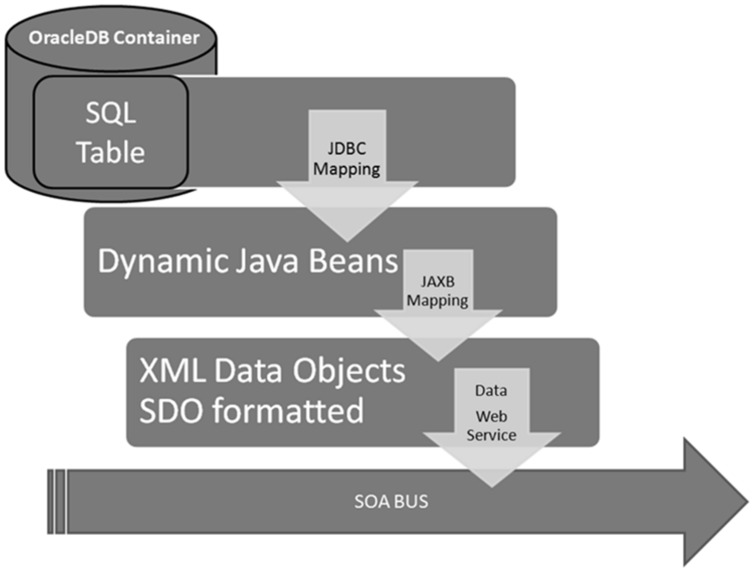

Fig. 6. SDO-based Web Data Service for Oracle Database

Using the Web Service model with the SDO standard implemented by the open source framework library of EclipseLink project, we have managed to ground a dynamic and agile data integration platform where the integration specific to the database federations have been addressed by turning object relational databases into Data Web Services, as in Fig. 6.
The SDO framework assumes self-describing XML data-sets containing inter-referenced data-objects that could be interchanged between data services in the context of SOA architecture. The XSD standard used to describe the SDO data sets is a natural fit to WSDL documents that contain the meta-data to describe interconnected Web Services.

Service discovery enhanced
Discoverability is one of the fundamental principles of service model (from SOA or from just plain Web Services Spec) having a major impact on *reusability* (developer does not need to build new services if there are other services providing the needed business functions or infrastructure functions). Also, we think that this principle has an overall impact on the *agility* of the dynamic

service architecture: *agile services* could *dynamic discover, interpret* and *dynamic-bind* to the resource-partner-services they need. This is a direct consequence of the separation of concern principle applied on SOA-WS systems so that every architectural component (Web Service) has to fulfill a clear delimited and modular function thus becoming a simple *link* that refers to other specific but distinct *linked functions* (from the Internet of Services). The standard Web Services specifications propose a protocol model based on service registries to store the WSDL-metadata needed to manage and dispatch the Web Services descriptions covering from real locations and declarative business functions (goals and requirements) to data-interchange protocols and invoking endpoints.

In our opinion, building, managing and promotion of this kind of service registries is a major bottleneck in the way of publishing and reusing Web Services in global architectures. The global architecture of Web Services could be a new Internet market for a new Internet business era where the very dynamic, virtual and collaborative enterprises will proliferate.

In this regard, we have started to develop an experimental proposal to use the existing searching infrastructure (meaning public search engines like Google and their enormous databases) as a huge and efficient *global web service search engine*. Our Web Service search (or discovery) protocol is based on these relatively simple but relevant principles and actions (sketched as a workflow in Fig.7):

1. *registering* every Web Service as (or into) a regular web application/website using a range of techniques starting from the traditional HTML meta-tags to the very sophisticated online marketing strategies like paid Add-Words (see https://adwords.google.com/) or Add-URL messages (see http://www.google.com/intl/com/add_url.html);
2. *searching* by using the specific search engine APIs, like Google Search API that could be invoked using Restful Web Services protocol. A search query could have a special dynamic subset of filters to get only relevant results containing:
   - URLs that reference to WSDLs;
   - web pages containing WSDL URLs or containing whole descriptors either as meta-information, either folded into special nodes (visible or not) of JSON documents which are parse-able using a JS/JQuery functions;
   - direct URL to Web Service endpoints ready to be invoked right away;
3. *parsing* the resultset of the search query processed by the search engine *and discovering* the relevant results that could contain Web Services descriptions or WSDL-URLs for Web Services matching searching criteria;
4. *interpreting* the Web Services standard descriptions from WSDL and
   - get the WS-endpoints with their operations that could be invoked;
   - get the WS-data-interchange-format, maybe from SDO-XSD documents;
5. *checking WS availability* and *binding WS-endpoint-URLs* to the data resource descriptors from a local service registry, in order to be re-checked and reused without having to go throughout the entire search and discovery cycle.

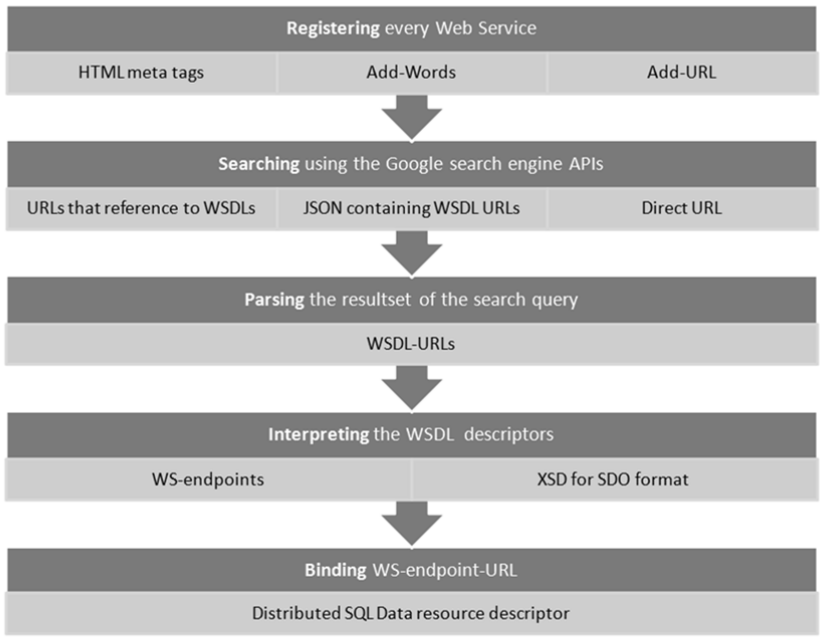
Registering every Web Service
HTML meta tags
Add-Words
Add-URL
Searching using the Google search engine APIs
URLs that reference to WSDLs
JSON containing WSDL URLs
Direct URL
Parsing the resultset of the search query
WSDL-URLs
Interpreting the WSDL descriptors
WS-endpoints
XSD for SDO format
Binding WS-endpoint-URL
Distributed SQL Data resource descriptor

Fig. 7. Discovery model based on GoogleSE

Distributed linked service queries
There are several theoretical and technological frameworks to process *distributed data resource queries* like DQP, Distributed Query Processor, of Open Grid Services Architecture-Data Access and Integration Services initiative [19], [23]. In this regard we have started our own experiment using Oracle APEX platform in order to ground a feasible framework to parse and execute OLAP distributed queries where dimensional data links represent the distributed data resources accessible as Data Web Services. We are building a technological extension to the already present Oracle SQL OLAP language that is natively accessible from the APEX environment. Our extension allows the distributed Web Services links to be directly referenced into SQL queries (FROM clauses) being static described into a local metadata repository or dynamically resolved using a technique inspired from *service discovery* framework based on SEO, as we have already described.

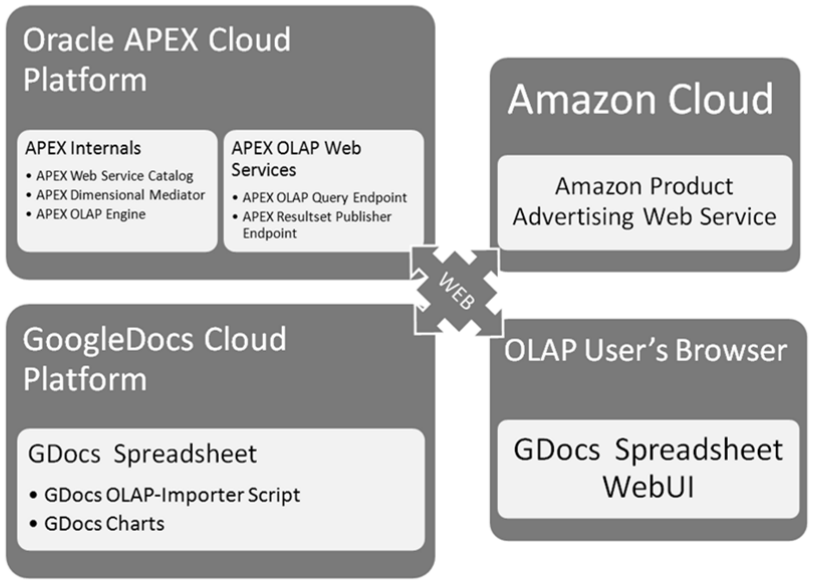
Oracle APEX Cloud Platform
APEX Internals
• APEX Web Service Catalog
• APEX Dimensional Mediator
• APEX OLAP Engine
APEX OLAP Web Services
• APEX OLAP Query Endpoint
• APEX Resultset Publisher Endpoint
Amazon Cloud
Amazon Product Advertising Web Service
WEB
GoogleDocs Cloud Platform
GDocs Spreadsheet
• GDocs OLAP-Importer Script
• GDocs Charts
OLAP User's Browser
GDocs Spreadsheet WebUI

Fig. 8. OLAP Web Service

This way could be developed an entire distinct Web Service computing model to build and to operate with a special type of Data Web Service, that we called OLAP-WebService as in Fig.8.

5 Conclusions

In this paper we have tried to analyze and to outline the defining characteristics of the current generation of Web Services and we have explored their "smart" opportunities for the upcoming potential Internet of services. We have identified a common service model of SOA, Web Services and Cloud Computing, which could enhanced the characteristics of its basic standards, specifications and platforms to describe truly dynamic, agile and autonomous Web Services.

We have explored some developing aspects of this kind of services, proposing some practical approaches like the dynamic discovering and linking protocol to achieve dynamic interoperability (and resource acquirement) between Web Services. In the near future we will start to experiment a method to dynamic deploy the Web Services into the global net of cloud platforms. Also, we intent to build a consistent and coherent development platform from these practical achievements, and to develop a set of measurements to assess the quality of these web services in order to further analyze their impact on the cloud service economic model.

Acknowledgements

This work was supported by the project "Post-Doctoral Studies in Economics: training program for elite researchers - SPODE" co-funded from the European Social Fund through the Development of Human Resources Operational Program 2007-2013, contract no. POSDRU/89/1.5/S/61755.

References

[1] T. Erl, *SOA Principles of Service Design*, Prentice Hall, Crawfordsville, Indiana, USA, 2008, ISBN-13: 9780132344821

[2] C.J. Date, *An Introduction to Database Systems (8th Edition)*, ISBN-13: 978-0321197849, Pearson Education, 2006

[3] M. Fotache, *Proiectarea bazelor de date. Normalizare si postnormalizare. Implementari SQL si Oracle*, ISBN 973-681-898-5, Editura Polirom, Iasi, Romania, 2006

[4] I. Ivan, C. Ciurea, M. Doinea, "Collaborative Virtual Organizations in Knowledge-based Economy", *Informatica Economica*, vol. 16, no. 1/2012, pp. 143-154

[5] L.G. CRETU, "Cloud-based Virtual Organization Engineering", *Informatica Economică* vol. 16, no. 1/2012, pp. 98-109

[6] R. Daigneau, *Service design patterns : fundamental design solutions for SOAP/WSDL and restful Web services*, Addison-Wesley, Pearson Education, Inc, Westford, Massachusetts, USA, 2012, ISBN-13: 978-0-321-54420-9

[7] P. Kuchana, *Software Architecture Design Patterns in Java*, AUERBACH PUBLICATIONS, 2004, ISBN 0-8493-2142-5, pp. 31-33

[8] J. Jarvis, *What Would Google Do?*, Publisher: HarperCollins Publishers, Inc.,ISBN: 0061726338, New York, USA, 2009

[9] G. Shroff, *Enterprise Cloud Computing: Technology, Architecture, Applications*, Cambridge University Press, Cambridge CB2 8RU, UK, 2010, ISBN 978-0-521-76095-9

[10] D. E.Y. Sarna, *Implementing and developing cloud computing applications*, CRC Press, Auerbach Publications Taylor & Francis Group, New York, USA, 2011

[11] B. Sosinsky, *Cloud Computing Bible*, Wiley Publishing, Inc. Indianapolis, USA, ISBN-13: 978-0470903568, 2011

[12] A. Edmonds, T. Metsch, and A. Papaspyrou, "Open Cloud Computing Interface in Data Management-Related Setups", in Sandro Fiore and Giovanni Aloisio editors, *Grid and Cloud Database Management*, Springer-Verlag Berlin Heidelberg 2011, ISBN 978-3-642-20044-1, Crawfordsville, Indiana, USA, 2009, pp.24-48

[13] V.F. Pais, V. Stancalie, "Using web services for remote data access and distributed applications", *Fusion Engineering and Design* 81 (2006), pp. 2013–2017

[14] S. Xiao, Z. Lin, G. Guang-hong, "Digital product data exchange in semantic service-oriented architecture", *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, Vol. 28 No. 6, 2009, pp. 1560-1578

[15] I. Saleh and G. Kulczycki, "Demystifying Data-Centric Web Services", *IEEE INTERNET COMPUTING*, SEPTEMBER/OCTOBER 2009, Published by the IEEE Computer Society, pp. 86-90

[16] S. Amer-Yahia, Y. Kotidis, "AWeb-Services Architecture for Efficient XML Data Exchange", *Data Engineering, 2004. Proceedings. 20th International Conference on*, IEEE Xplore, 2004, pp. 523-534

[17] V. Borkar, M. Carey, N. Mangtani, D. McKinney, R. Patel and S. Thatte, "XML Data Services", *International Journal of Web Services Research*, 3(1), 85-95, January-March 2006, IDEA GROUP PUBLISHING, Hershey, USA

[18] T. Erl, A. Karmarkar, P. Walmsley, H. Haas, U. Yalcinalp, C. K. Liu, D. Orchard, A. Tost, J. Pasley, *Web Service Contract Design and Versioning for SOA*, PRENTICE HALL, ISBN-13: 978-0-13-613517-3

[19] S. Lynden, O. Corcho, I. Kojima, M. Antonioletti, and C. Buil-Aranda, "Open Standards for Service-Based Database Access and Integration" in S. Fiore and G. Aloisio editors, *Grid and Cloud Database Management*, Springer-Verlag Berlin Heidelberg 2011, ISBN 978-3-642-20044-1, Crawfordsville, Indiana, USA, 2009, pp. 3-21

[20] B. Sosinsky, *Cloud Computing Bible*, Wiley Publishing, Inc. Indianapolis, USA, ISBN-13: 978-0470903568, 2011

[21] C. Mayr, U. Zdun, S. Dustdar, "View-based model-driven architecture for enhancing maintenability of data access services", *Data & Knowledge Engineering*; 70-2011; 70; p.794-819.

[22] L. Resende, "Handling heterogeneous data sources in a SOA environment with service data objects (SDO)", *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, ACM, New York, NY, USA, 2007, pp. 895–897

[23] B. Dobrzelecki, A. Krause, A. Hume, A. Grant, M. Antonioletti, Y. Alemu, M. Atkinson, M. Jackson, E. Theocharopoulos.: *Integrating distributed data sources with OGSA–DAI DQP and Views*. Phil. Trans. R. Soc. A 368(1926), 4133–4145 (2010)

Cătălin STRÎMBEI has graduated the Faculty of Economics and Business Administration of Al. I. Cuza University of Iaşi in 1997. He holds a PhD diploma in Cybernetics, Statistics and Business Informatics from 2006 and he has joined the staff of the Faculty of Economics and Business Administration as teaching assistant in 1998 and as senior lecturer in 2005. Currently he is teaching *Object Oriented Programming*, *Software Development Environments* and *Database Design and Administration* within the Department of Business Information Systems, Faculty of Economics and Business Administration, Al. I. Cuza University of Iaşi. He is the author and coauthor of four books and over 25 journal articles in the field of object oriented development of business applications, databases and object oriented software engineering.