# A SCIENTIFIC RESOURCE SPACE MANAGEMENT SYSTEM

*Cristhian Parra, Marcos Baez, Florian Daniel, Fabio Casati, Maurizio Marchese*
University of Trento – Dipartimento di Ingegneria e Scienza dell'Informazione
Via Sommarive 14, 38100 Trento, Italy
{parra,baez,daniel,casati,marchese}@disi.unitn.it,

*Luca Cernuzzi*
Universidad Católica "Nuestra Señora de la Asunción" – Departamento de Electrónica e Informática
Tte. Cantalupi y Tte. Villalón. Barrio Santa Ana. Asunción – Paraguay
lcernuzz@uca.edu.py

February 2010

Technical Report # DISI-10-013
Version 1.0

# A Scientific Resource Space Management System

Cristhian Parra, Marcos Baez, Florian Daniel, Fabio Casati, Maurizio Marchese
University of Trento – Dipartimento di Ingegneria e Scienza dell'Informazione
Via Sommarive 14, 38100 Trento, Italy
{parra,baez,daniel,casati,marchese}@disi.unitn.it,

Luca Cernuzzi
Universidad Católica "Nuestra Señora de la Asunción" – Departamento de Electrónica e Informática
Tte. Cantalupi y Tte. Villalón. Barrio Santa Ana. Asunción – Paraguay
lcernuzz@uca.edu.py

## ABSTRACT

As the web continues to change the way we produce and disseminate scientific knowledge, traditional digital libraries are confronted with the challenge of transcending their boundaries to remain compatible with a world where the whole Web in itself is the source of scientific knowledge. This paper discusses a resource-oriented approach for the management and interaction of scientific services as a way to face this challenge. Our approach consists in building a general-purpose, extensible layer for accessing any resource that has an URI and is accessible on the Web, along with appropriate extensions specific to the scientific domain. We name the class of systems that have this functionality Scientific Resource Space Management Systems, since they are the resource analogous of data space management systems known in literature. In this paper, we describe the motivations, concepts, architecture, and implementation of the platform and one validating usage scenario.

## Categories and Subject Descriptors

H.3.5 [**On-line Information Services**]: Web-based services. H.3.7 [**Digital Libraries**]: Dissemination, Collection. D.2.11 [**Software Architectures**]: Service-oriented architectures. H.3.3 [**Information Search and Retrieval**]: Search process, Selection process.

## General Terms

Management, Documentation, Design

## Keywords

Resource Space Management System, Scientific RSMS, Scientific Resources, Karaku, ResMan

## 1. INTRODUCTION

Over the last decade, the increasing outburst of services available on the Web has pushed forward *new ways of producing and disseminating knowledge online*. For instance, in the context of scientific knowledge today's researchers have access to an overwhelming space of scientific publications thanks to instruments that range from traditional digital libraries (such as SpringerLink[1] and ACM[2]) to specialized search engines (such as GoogleScholar[3]) and metadata services (such as DBLP[4]). But not only: in addition to these rather "traditional" means of knowledge dissemination, today's Web 2.0 is characterized by instruments that provide for the *earlysharing* of knowledge, e.g., wikis, blogs or personal web sites. Even though these kinds of contributions are not peer-reviewed, depending on the reputation of the author, they might nevertheless have a huge impact on the scientific community (think, for instance, of the so-called technology evangelists). Then, there is an increasing interest in online repositories where scientists can publish, share and discuss their contributions, leveraging the power and typical features of *social applications*. For instance, scientists might share and collaboratively enhance teaching material like slides, books or videos or they might share their data and experiments like, for instance, on myExperiment.org where scientists share their scientific workflows.

These latter, novel kinds of contributions, however, are typically not considered *first-class citizens* in scientific knowledge dissemination. Yet, we argue that in many cases they provide significant contributions to science that complement the traditional research papers. As such, they too need to be properly indexed and made available to the public for search and access, a task that is however not easy. The biggest hurdle we need to clear is the *heterogeneity* of these contributions. In fact, unlike in digital libraries where there exist efforts toward the definition of standard means (e.g., interfaces, languages, protocols) to access and query online repositories, wikis, blogs or social applications typically do not feature similar interfaces. Rather, they follow the recent trend of exposing software interfaces on the Web, such as SOAP or RESTful web services, which can be used to programmatically interact with them. Unfortunately, however, there are no standards for the design of these kinds of web-accessible APIs and, as a consequence, there is no single instrument to search and access these heterogeneous sources in a uniform fashion, and each source requires its own access logic.

Enabling users to search the described types of scientific contributions therefore requires a novel approach, especially as for what regards the logic of *how to access* individual sources (multiple

[1] http://www.springerlink.com

[2] http://portal.acm.org
[3] http://scholar.google.com
[4] http://dblp.uni-trier.de

technologies might be involved in a single query) and of *how to abstract* them to the user (who doesn't want to know about the technicalities). The goal of this paper is to extend the reach of search services, such as the ones supported by traditional digital libraries, beyond their typical boundaries. For this purpose, we leverage on ideas taken from dataspace management [1] so as to develop what we call a *Scientific Resource Space Management System* (sRSMS), which will allow us to access a variety of scientific resources homogeneously, addressing the problem of *heterogeneity* and *interoperability* among scientific artifact sources (not only digital libraries) in a novel fashion and enabling the easy development of *value-adding applications* on top.

In essence, our sRSMS provides homogeneous programmatic access to scientific resources by (i) abstracting the various kinds of *scientific knowledge* into a uniform conceptual model to support uniform access logics; (ii) abstracting the *operations* that the services providing access to scientific knowledge support (from simply accessing paper data and metadata, to extracting and tagging content, crawling citations, accessing blogs and experiments, changing access rights, posting, submitting for review, etc…); and (iii) by hiding the tedious problem of accessing *heterogeneous platforms*, which very often are not even available for programmatic access but are only designed for web browser access (e.g., SpringerLink, Google Scholar, blogspot, or wikis).

**Motivating scenario**. The idea of sRSMS was born in the context of the EU project *Liquidpub*[5], which aims at developing concepts, models, metrics, and tools for an efficient (for people), effective (for science), and sustainable (for publishers and the community) way of creating, disseminating, evaluating, and consuming scientific knowledge. For this purpose, several tools are under development, providing advanced features on top of what we call the scientific resource space. Among them, we aim, for instance, at developing so-called *Liquid Journals* (LJs), i.e., personal collections of scientific resources (the journals) that evolve continuously over time, following the dynamics of the resources it is built on (the liquid aspect). For this purpose, it is necessary to query both traditional, peer-reviewed journals and conferences, and the novel kinds of contributions discussed above. In Figure 1 we illustrate the idea that drives this paper: for the purpose of fast prototyping and early validation of the LJ idea, we started implementing the LJ Application as a monolithic block, which indeed allowed us to achieve the expected results in short time. However, we also recognized that there is something more "under the hood", which deserves its own attention, especially in light of other advanced features to be implemented: the abstraction and management of the actual scientific resources.

Providing these features in a way that is as general and widely applicable as possible and, at the same time, as useful and specific (to the scientific domain) as possible is a non-trivial task. There are several challenges that need to be answered: Which are the best concepts and abstractions? Which features are general enough to be really reused in practice? How does our resource space look like? How do we deal with the heterogeneity of resources? How do we query the resource space? Which interfaces are suitable to provide programmatic access to application to be developed on top? And so on.

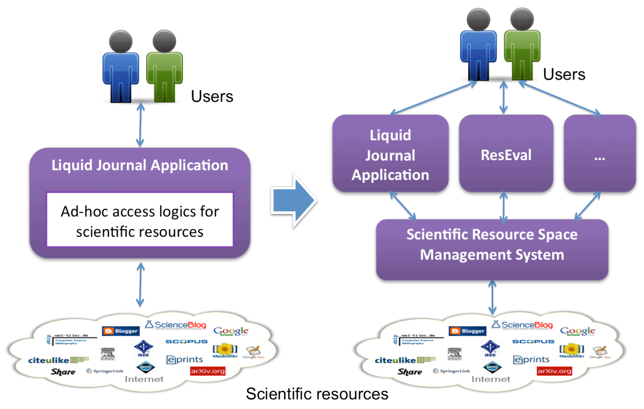
Users
Liquid Journal Application
Ad-hoc access logics for scientific resources
Users
Liquid Journal Application
ResEval
...
Scientific Resource Space Management System
Internet
Scientific resources

**Figure 1 From ad-hoc access of scientific resources to a dedicated Scientific Resource Space Management System**

**Contributions**. Building on this scenario, in this paper we provide the following contributions:

- We introduce the ideas of Resource Space Management System (RSMS) and **Scientific Resource Space Management System** (sRSMS) and describe the ideas and requirements that drive their development.
- We define our **scientific resource space** and show how to abstract scientific resources of various natures along with their operations. While anything identified by a URI is in scope of a generic RSMS (whether it is a scientific resource or not), here we aim at providing concepts and services for scientific resources, such as built-in notions of authors, references, and the like. Although tailored to the domain of scientific publications, we will see that the introduced concepts are extensible and easily applicable also to other domains.
- We describe the **implementation of our sRSMS**, which is able to provide homogenous programmatic access to resources and web services, regardless of how they are implemented as long as they are web-accessible.
- We show how our sRSMS can be **leveraged to ease the development** of the Liquid Journal Application described above and show how it similarly supports the development of a prototype tool for the evaluation of a scientist's scientific impact on the community (see the ResEval application in Figure 1).
- Finally, in doing so we aim at **simplicity, flexibility and collaborative extensibility**. Our target users are people with skills comparable to using spreadsheets or wikis. So, simplicity is paramount if we want to support them in their tasks. Yet, more expert users (e.g., other developers) might feel the need to extend our sRSMS with own logics. Given the large amount of services available, it is practically impossible to provide a monolithic infrastructure that incorporates all of them. It is also unfeasible to think that we can develop (or even envision) all services that will use the sRSMS or even that can be part of the core sRSMS. Our sRSMS therefore facilitates extensibility by the community in that developers can just register services that interface with systems or scientific resources and that may be hosted also by other parties (there is no need for plugging code in).

[5] http://project.liquidpub.org

## 2. RELATED WORK

Our idea of resource space management system stems from the area of *dataspaces*, which extends concepts from traditional database management toward heterogeneous data sources [1][2]. In particular, the idea of dataspaces is to allow building applications over a dataspace layer by searching for and operating with multiple data sources using a common interface. We extend the principles of dataspaces to a *space of scientific resources*, where resources also have own behaviors (i.e., they have actions that can be used to interact with them), and we aim to model scientific entities in this space of resources as first class citizens. We differ from dataspaces in that we do not only focus on the search problem, but we also provide abstractions for operating with scientific entities at different levels of abstraction/granularity.

If we look at the Web, we see that electronic publishing, digital libraries, electronic proceedings, on-line patents repositories and more recently blogs and scientific news streaming are rapidly expanding the amount of available scientific/scholarly digital content.Search engines (like Google, Yahoo, Ask.com, and so on) give users a first, shallow (but easy to use) level of integration through keyword-based search, and have an excellent coverage of material published on the Web. The introduction of smart ranking algorithms, such as Google's PageRank™, made this type of search even more effective and fast. However, keyword-based search has some heavy limitations, such as: document-level granularity, lack of integration across results, lack of context for keywords, difficulty in expressing complex queries (for example, one cannot directly ask Google a query like "*Give me all publications written between 2000 and 2009 by authors who have an h-index above 10 in the specific research domain of interoperability in Digital Libraries*"). The problem is that general-purpose search engines lack the necessary domain concepts and interaction capabilities to properly handle scientific resources.

The Search Computing[6] (SeCo) project aims at answering queries that are similar to the one above by automatically deriving from the query in input a suitable query execution plan, which can then be used to orchestrate the interaction with individual search services [3]. The goal of the project is to enable so-called multi-domain search, i.e., the search of deep web data by accessing multiple domain-specific search services in a coordinated fashion. In order for search services to be accessed by the SeCo query engine, they must have been registered in the platform. Resources (the search services) are however interpreted as pure search services, while in our sRSMS (i) we do not address only search services, but also single scientific resources (e.g., an individual Google Doc); (ii) we aim to handle scientific resources like full-fledged services with their own interaction logic; and (iii) we try to provide the resources' features to upper layers in the software stack in an abstract form.

Indeed, our research also considers the problem of operating on sources (not only search). Thus, a relevant area is that of services integration and interoperability, where research on service compatibility [4], and recently on models and frameworks for service integration, replaceability, and interoperability has produced results we build on in our work [5][6].

Besides general-purpose search engines, there are many open or commercial digital libraries that specifically focus on the scientific knowledge domain, such as Scopus[7], Web of Knowledge[8], CiteSeer[9], DBLP, or GoogleScholar, which typically offer a much better and more flexible access to their content. Flexibility and search effectiveness derive from the use of annotations of contents and documents with structured metadata. As a consequence, they can answer queries that are more complex than simple keyword queries (in principle, even to the query specified above). However, they suffer from a much narrower coverage, and currently there is very little – if any – integration between different services. This means, for example, that DBLP or CiteSeer cannot answer any query that requires gathering information from each other or from related digital libraries like the patent library of EPO or the project database of CORDIS[10].

In the context of scientific knowledge, the challenge is therefore developing applications that are capable of using these repositories to assist the scientific community above and beyond the pure dissemination of information. One important thread of work is related to the definition of standards for metadata for scientific/scholarly content in order to support this kind of integration. In particular:

- The *Dublin Core Metadata Initiative*[11] (DCMI) is dedicated to promoting the widespread adoption of interoperable metadata standards and developing specialized metadata vocabularies for describing digital resources.
- *OpenArchives Initiative Protocol for Metadata Harvesting*[12] (OAI-PMH) defines a mechanism for harvesting records containing metadata from repositories. Thus, metadata from many sources can be gathered together in one database, and services can be provided based on this centrally harvested or "aggregated" data.
- *Online Information Exchange*[13](ONIX): is the international standard for representing and communicating book industry product information in electronic form.
- The *Digital Object Identifier*[14] (DOI®) serves as an identifier for digital objects on digital networks. It provides a system for persistent and actionable identification and interoperable exchange of managed information on digital networks.

All the above protocols, however, are focused on a top-down approach for supporting content interoperability (metadata from repositories), which is only one angle of the problem we are facing in our approach. More specifically, it misses on recent bottom-up trends of exposing scientific artifacts (not only papers) via software interfaces on the Web, which can be used to programmatically interact with them. In this respect, it is worth mentioning recent services, such as CiteUlike[15], SciLink[16], SciSpace[17], Mendeley[18], or Zotero[19], which provide scientists with the tools

[6] http://www.search-computing.it/

[7] http://info.scopus.com/

[8] http://isiwebofknowledge.com/

[9] http://citeseer.ist.psu.edu/

[10] http://cordis.europa.eu/home_en.html

[11] http://dublincore.org/

[12] http://www.openarchives.org/OAI/openarchivesprotocol.html

[13] http://libraries.mit.edu/guides/subjects/metadata/standards/onix.html

[14] http://www.doi.org/

[15] http://www.citeulike.org/

[16] http://www.scilink.com

[17] http://www.scispace.com/

[18] http://www.mendeley.com/

for organizing and sharing papers, creating social communities, and making contacts. However, these services, although very interesting and useful, do not provide any reusable infrastructure to build new applications on top. They can however be seen as a new type of scientific resource that can be used by our sRSMS infrastructure.

## 3. RESOURCE SPACE MANAGEMENT: CONCEPTS AND REQUIREMENTS

Before introducing our interpretation of sRSMS, it is good to clarify which are the theoretical assumptions and design principles that drive the development of our sRSMS and that, we believe, should be common to every RSMS, be it scientific or not. In this section, we leverage on the ideas pushed forward in [1], where the authors introduce the concept of *Dataspace Management* and *DataSpace Support Platform* (DSSP) in the context of data management, and we describe our analogies in the context of resource management. For instance, our idea of RSMS can be seen as resource counterpart of the DSSP.

Managing a space of resources means bringing together inside one homogeneous environment a variety of heterogeneous kinds of resources and providing suitable means to access and use resources and to define and maintain all necessary relationships among the resources. In short, a *resource* can be any artifact we can refer to by a URI and that is accessible over the Web. This notion is very general and captures the requirement of supporting any arbitrary information such as simple web pages, online documents, web services, feeds, and so on. That is, resources might be simple sources of data or content, but they might also be as complex as SOAP or RESTful web services with their very own interaction logic. One of the main challenges in developing a RSMS is exactly to hide all this complexity and heterogeneity that might characterize the resources to be managed.

A *resource space* can then be defined as a set of resources and relationships, where the set of resources limits the space to a manageable number of resources, and the relationships express how the resources in the space are interrelated. Theoretically, the biggest resource space with our definition of resource is the Web itself, but, of course, we do not aim at providing a new way of managing the Web. Instead, we think that only by setting suitable boundaries for the resources to be considered, i.e., by limiting the resource space, it is also possible to provide value-adding, novel functionalities that justify the development of a dedicated RSMS. For instance, in this paper we focus our attention to the specific domain of scientific knowledge.

Given a resource space, *resource space management* means providing, on top of the resource space, functionalities that allow one (either programmatically via suitable programmable interfaces or via human interaction) to organize the space and to handle its resources, making the most of their individual capabilities. Such functionalities are to be enabled by the RSMS, of which we specifically identify the following *services* as basic features (adapted from [1]):

- **Cataloging** of resources and of the content and services that are accessible through those resources: This is the first and foremost service of a RSMS. The catalog is the instrument that allows one to define the actual resource space and to limit it to a manageable size. Cataloging resources therefore means (i) defining the nature and capabilities or resources, (ii) specifying and maintaining relationships among the resources, (iii) storing and indexing the resources in the catalog, and (iv) managing the metadata that are necessary to configure the resources in the resource space for access and interaction.
- **Querying/Searching** the resource space: Once a resource space is defined, in order to provide access to its resources it must be possible to query or search for resources. With *querying* we refer to exactly answering structured queries over the resource space, analogously to how we query a relational database. With *searching* we mean search in terms of keyword-based, unstructured queries, analogously to how we query the Web.
- **Supporting complex workflows** over resources in the resource space: Some maintenance operations or application features on top of the RSMS might require the execution of coordinated actions over resources in the space. Such feature could be, for instance, achieved by supporting workflows of operations over the resources or compositions of web service interactions.
- **Monitoring and handling events**: As resources are not static and evolve over time – especially on the Web where not only contents but also programmatic interfaces and, hence, the features provided by the resources typically change at a fast pace – it is important to keep the local description of the resources in the catalog up to date. Depending on the nature of the resources, it might be possible to monitor their evolution (e.g., via events emitted by the resources) or it might be necessary to query them for updates.
- **Analyzing resources and the resource space:** Managing a resource space means understanding the health of the space and taking actions in case of problems. Doing so requires the RSMS to provide basic analysis features that inform about the state of the space. The supported analysis features may vary depending on the type of resource and resource space supported by the system.
- **Discovering** of resources in the resource space: Next to managing the dynamics of the resources in a resource space, it is also necessary to manage the dynamics of the resource space itself, since on the Web continuously new resources are created and others are destroyed. It is therefore also important to be able to discover (e.g., by crawling the Web) those new resources that satisfy the membership requirements of the resource space and to add them to the space, allowing the space to grow autonomously.

Ideally, a RSMS supports all of the above features, plus additional ones that vary depending on the specific application domain they focus on. In practice, already a subset of the above features may provide substantial help to its users, especially if – in addition to the pure management of resources – the system also provides effective instruments that allow the user to handle resources and to interact with them at the level of abstraction that best suits the chosen domain. In the next section, we show how this additional layer could look like if we focus on the scientific knowledge domain; then we explain how resource management in the resulting sRSMS is supported by our underlying RSMS.

[19] http://www.zotero.org/

## 4. MANAGING SCIENTIFIC RESOURCES

A generic RSMS as discussed in the previous section allows us to interpret the Web as resource space in which all URI-accessible artifacts are resources. The goal of this paper is to go beyond the mere technology abstraction and to also provide suitable domain concepts that not only simplify the access of and interaction with resources, but also represents them in a way that can be understood by non-IT people and domain experts. Doing so will allow us to widen the accessibility of our sRSMS from IT experts to average web users. In this section, we show how we achieve this in our sRSMS called *Karaku*[20].

We have seen that the notion of resource is very general and captures the requirement of supporting any arbitrary resource on the Web. If we refine the idea of resource in the context of scientific knowledge, we can define a *scientific resource* as any resource that represents an important concept in the domain of scientific knowledge dissemination. For instance, documents (e.g., a Google Doc or a blog entry), experiments results in the form of datasets, metadata information like DBLP's records about scientific publications and authors, authors themselves, and so on can be seen as resources. A *scientific resource space* is therefore the space that contains all the scientific resources we think are necessary to describe the space.

In order to develop our sRSMS, we have followed a two layered architectural design, where the upper layer provides the set of functionalities that are domain-specific (domain specific catalog and caching, query support, monitoring and crawling), and the base layer provides a set of core services to support the management of generic resources (catalog, access and interaction). Next we show how we specify the scientific resource space in Karaku and also explain Karaku's high-level architecture. Then we focus on the actual resource management.

### 4.1 Modeling the Scientific Resource Space

In order to support and push forward a group of innovative scientific services, the first step is to speak the same language used in the domain of scientific research. The first step is therefore to define a comprehensive conceptual model that supports all possible entities and relationships in the specific domain that will be common for all services built upon this layer. Although many initiatives are being done to come up with such a model (e.g. OAI-MPH), none of them have had an impact enough as to become the industry "de facto" standard. In this paper we therefore introduce our own model, which is tailored to the specific features we want to support in the sRSMS. But before proceeding with the description of the actual model, we describe the modeling formalism (i.e., the meta-model) Karaku understands for the definition and operation of the scientific resource space.

Specifically, we can model the scientific resource space by means of three basic constructs (very similar to the well-known Entity-Relationship notation):

- **Entities**: entities define the domain concepts we want to manage in the sRSMS. Entities are the domain-specific representation of the resources available on the Web and, as such, can be characterized by means of a name, properties, and possible operations (i.e., actions) that can be performed on the resource.
- **Relations**: relations (or relationships) define connections between two different entities (e.g., cited by, coauthored with, it is affiliated to). Relations are at the basis of query evaluation and allow the query engine to relate different entities.
- **Annotations**: annotations represent extended information attached to both relations an entities (e.g., comments, specialized attributes like tags or similar). Annotations can be used to improve search performance and to specify how to bind entities to actual concepts.

[20] http://project.liquidpub.org/karaku/. Karaku is a Guaraní word traditionally used to refer to the core of an issue. It was chosen as the name of this project because it is the core element in the overall Liquidpub project platform.

The former two constructs allow us to model the scientific resource space. Via annotations, it is possible to extend the scientific entities by adding more attributes and technical details. Indeed, we can think of the space of scientific resources as conglomeration of resources being tagged with different "types", relationships, and allowed actions. Defining such a classification allows us to manage resources more easily while also providing guidelines for further specializations.

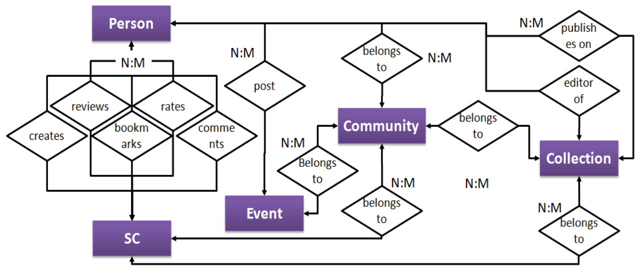

**Figure 2. Conceptual model of the scientific research space**

Figure 4 shows the model of the scientific resource space we currently support in Karaku. Boxes represent entities, connectors represent relations, and rhombuses label the relations. The figure does not render annotations, which we skip for presentation purposes. The entities we want to manage are:

i) **Scientific contributions** (SC): these represent the actual scientific knowledge artifacts, such as papers, reviews, blog entries, experiments, etc. The scientific contributions are the main entity around which we define the other four entities.

ii) **Person**: scientific contributions are produced by people, which we represent by means of the Person entity. Depending on their involvement in the knowledge production process, people may play different roles from the perspective of the scientific contribution, which we represent by means of suitable relationships.

iii) **Communities**: communities refer to groups of people working in a same field or area of research. Knowledge about the communities and the involvement of people in communities is particularly interesting if we want to assess the "quality" or impact of a researcher within his community. Communities typically evolve frequently over time.

iv) **Collections**: collections are predefined aggregations of both people (e.g., institutions) and scientific contributions (e.g., conference proceedings). Typically, collections are persistent in time or change only at a very slow pace.

v) **Events**: events are occasions taking place at a particular time, e.g., conferences, meetings, workshops, etc. bringing together people for discussion and publication of scientific results.

Here we concentrate on the definition of the scientific resource space. In the next section, we map these scientific entities to the resource space by defining suitable resource types that encapsulate the properties (and metadata), states and behavior of the above scientific entities. The properties are type-specific and allow us to define the information that characterizes particular instances and also their relations with other resources.

Regarding the model in Figure 2, the essence is not just the model in itself (although we found this simple model fits our needs fairly well, it is possible to argue that others are as good) but the fact that it can be extended or even replaced by another in the same sRSMS architecture (by means of the three introduced modeling constructs), offering in this way an opportunity to explore the concepts that form the foundations of scientific activity.

It is also important to notice that any domain can develop its own RSMS based on the same high-level constructs and the basic access layer that is discussed in this paper. The scientific community could even develop a new scientific RSMS, much more complex and expressive than the one we describe in this paper.

### 4.2 Karaku Architecture

**Given the above characterization of our scientific resource space, we need also a number of services to interact with it. In**

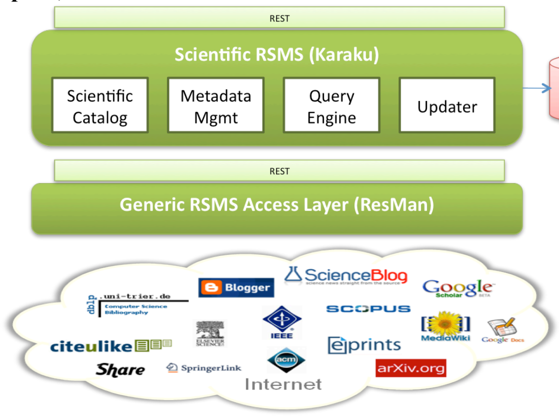

Figure 3 we show the overall architecture of our platform, including the following functional components:

i) **Scientific Catalog**: locally stores the above model of the scientific resource space, along with the necessary annotations.

ii) **Query Engine**: provides the mechanisms to answer the queries of the clients, expressed in a domain-specific query language expressed over the scientific catalog. Thanks to this module, upper layers will have access to different resources, regardless of the specificities of the source, by the means of queries like "*Get Contributions of Person X where Topic is equal to Y*" or "*Get Top-K Contributions of Collection Z*". The scientific resource space would be useless without a well-designed query language to take advantage of it.

iii) **Metadata Management**: provides the basic CRUD functionalities over the resources expressed in terms of the proposed conceptual model.

iv) **Updater**: provides capabilities to pull in metadata from the underlying RSMS, in order to populate and keep updated the locally cached metadata, used for efficient query processing.

In the current version of the prototype, the Metadata Management component has been fully implemented and tested; it is exposed as RESTful web service API. The other components are at the prototype level. It is worth to mention that in the whole project's design and implementation we have followed a *resource oriented architecture* approach [7] to be compliant with the latest tendencies on web services development.

Using the RESTful API provided by the query engine, a client can execute simple queries in the form of HTTP operations over the components in the model (e.g., it is possible to retrieve the list of all contributions by means of the following HTTP GET request: *http://project.liquidpub.org/karaku/contribution.xml*).

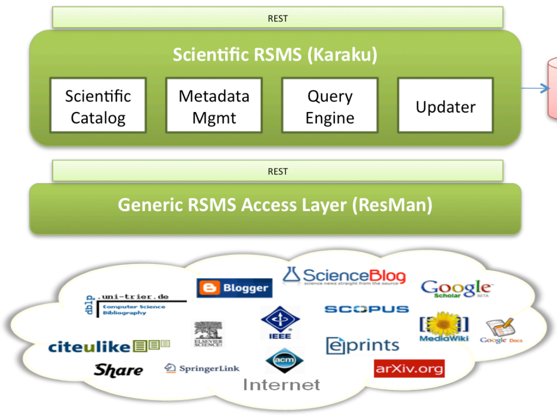

**Figure 3. Architecture of the Karaku sRSMS**

All these components provide a common model for the resources in the focused domain according to the model introduced in Section 4.1. Yet, we still have to face the problem of accessing the actual resources in the resource space. For this purpose, we rely on the *Generic RSMS Access Layer* shown in Figure 3 and described in the following.

## 5. TRANSPARENT ACCESS TO RESOURCES ON THE WEB

The access layer of our RSMS provides us with abstractions for modeling the vast amount of resources the Web offers and allows us to take into account also the software aspects involved in accessing the resources. Indeed, the huge variety of resources that can be part of our sRSMS is managed by different service providers that may or may not have an API (e.g., Google Docs, various flavors of wikis, Flickr, Google Scholar, etc). We refer to these service providers as *resource managers.*

In the scenario depicted by resources and resource managers in the Web era, it is not trivial to provide abstractions, given the heterogeneity in the resource managers. For instance, in Figure 4 we illustrate such heterogeneity showing some examples of currently available and relevant scientific domain services, all of them providing access to their content (the scientific contributions) via different APIs/protocols.

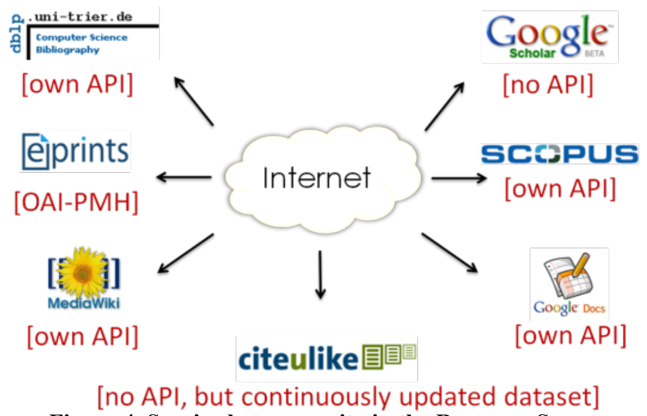

**Figure 4. Service heterogeneity in the Resource Space**

The reason for separating our general model in two layers is mainly the applicability. In the upper layer we focus on the requirements of the scientific domain, to provide a support platform for services that need to access scientific resources. The concepts used in the Access Layer are instead general and could be used in any other domain.

In the following we discuss the details of our access layer middleware, *ResMan*[21] **Error! Reference source not found.**, whose main function is to abstract the technical *access-to-resource* specifics, providing for them a universal resource space access layer. We also include a discussion of the prototypal implementation of the component.

## 5.1 The Basic Resource Space

In Section 4.1, we have introduced the model of the scientific resource space we support in Karaku, and we have explained that each of the entities can be annotated with meta-data that allows us to bind the entities with resources types, i.e., with a physical access mechanism. In this section, we look at how we bind resource types to actual resources. Figure 5 show the conceptual model that introduces the necessary concepts and that is also the relational model for the *resource space catalog* of ResMan.

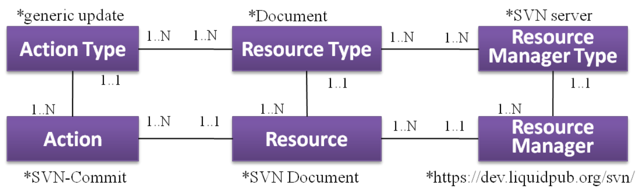

**Figure 5. Resource space conceptual model**

The first two elements of our model, *Resources* and *Resource Manager,* have already been introduced in Section 4. To illustrate their difference, let's consider two examples: a specific scientific article in *Google Scholar* is a resource while the *Google Scholar* service is the resource manager. In principle, there are no limitations for the kind of resource managers we can support, as long as they provide services for resources. Indeed, the third element we consider is the service or *action*. Actions describe the services provided by resource managers and that allow us to operate with the resources (e.g., to share, publish or search documents, or more complex actions such as crawling a web site such as scholar or Citeseer for scientific metadata).

At the level described above, the basic elements provide operations and properties, which are specific to the actual resource managers. For example, operating on a Google Scholar indexed article will be constrained to the set of Google Scholar-specific actions, these actions signatures and formats. Therefore, to free upper layers of implementing resource managers-specific operations, we provide a set of abstractions on top of these basic elements.

Incidentally, these abstractions are natural extensions of the basic elements. Thus, the first abstraction we consider is the *resource type*, which characterizes families of resources with similar behavior. For example, all the documents from Google Docs are of the type "Google Doc Document", documents stored in a SVN repository are of the type "SVN document" and if we consider a higher level of abstraction we can say that documents from both resource managers are of the type "Document". This idea can also be applied to resource managers, so we can group them into *resource manager types* to denote general classifications such as repositories, search engines, control version systems, etc.

Then, it is also the case that, even though the managing application is different, the kinds of actions that can be executed on the resource are similar. For example, in both Wiki and Google-Docs we can have the possibility of changing the access rights, publishing, etc. Some of these actions are semantically equivalent but may require different parameters (i.e., the "signature" details are different). We include in our model the *action type* abstraction as a way of providing a common interface for these semantically equivalent actions. In doing so, we can provide homogenous access to resources supporting the action-type. Finally, the model of resource space presented here will allow us to manage arbitrary resources at different levels of abstractions using a homogeneous interface.

## 5.2 ResMan Architecture

The universal RSMS access layer builds on the model introduced in the previous section and provides seamless access to resources disseminated over the Web. As depicted in Figure 6, the RSMS universal access layer architecture is composed of two main modules: the *resource space management* and the *access management modules*. These two modules run the machinery for providing homogeneous access to resources and transparent extensibility in terms of multiple resource managers' support.

[21] http://project.liquidpub.org/resman

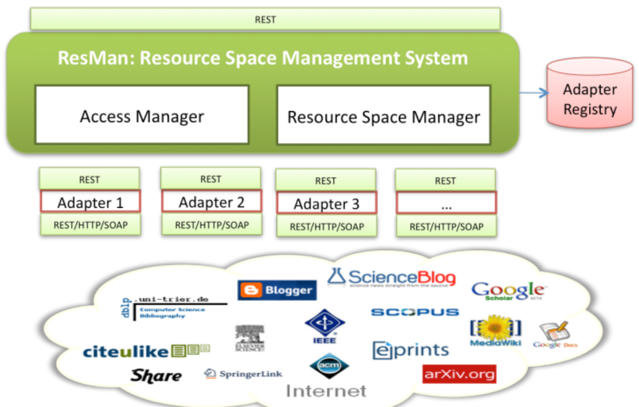

**Figure 6. RSMS architecture**

**Resource space management**. The resource space management module allows extending the resource managers (repositories, search engines, blogs, etc) available to the upper layers. Thus, this module allows us to *register* resource managers and the related *resources* and *actions*. It also manages the *mapping* between these constructs and the abstractions of resource types, action types and resource manager types. The link between the actual resource managers and the abstractions we provide is performed through *adapters* [6], described in the next subsection.

The registration or resource managers is performed using a specialized service that enables resource manager providers and programmers to populate a registry of resource managers and to make them available to upper layers. Note that it is also possible to define and register *composite* resources by combining actions from different resources into a complex resource type. This is particularly interesting for applications in which the conceptual resource can be composed of multiple low level artifacts (e.g., a virtual folder that contains elements which are references to Google docs, Zoho, or MS Word documents stored in an SVN). From the perspective of a client using the module, this acts as a "dictionary" that offers information about the resources, actions, resource managers and their abstractions, available in the registry.

**Access management**. The access management module allows interfacing with different repositories and libraries through a standard interface. This module is able to operate on resources of the same type (e.g. documents) with the same set of operations (e.g., create, delete, share) using the *resource-type* level of abstraction. In other words, this module allows executing actions on the resource managers registered from the resource space management module. Note that this is different from executing operations directly on the adapters where one can perform operations only on actual *resources*, and so the set of operations available are specific to those specific resources. For example, consider executing the operation "sharing" over a set of resources provided by different resource managers. The actual implementation of the action "share" will likely have a different signature in each adapter. The access module abstracts these differences allowing clients to operate at the action type level of abstraction, which in this example will be the "share" action type.

As illustrated in Figure 7, the interaction with the resource managers (the services providers) is performed through adapters. The Access Management module interfaces with the adapters and exposes their functionalities to the upper layers. The added value here is the possibility of working with different resources managers at a different level of abstraction; i.e., clients of this module do not need to know the details of the actual resource managers, indeed, they do not need to know which resource manager is providing a given service. The access management module, according to the specification of the resource types, manages this interaction.

## 5.3 The role of adapters

The approach we follow to guarantee extensibility, interoperability and maintainability is to provide a set of core modules that can manage the *adapters* and access to resource managers through these adapters. Each adapter provides a definition of the resources and operations supported and, if necessary, the implementation of the logic for accessing the resource managers (e.g., in case no API is provided). Figure 7 illustrates how the interaction with the adapters is performed.

Adapters are provided by third parties and made available to the upper layers through the resource space management module, which adds the adapter to the registry of adapters. Note that the approach we take here allow us to extend the services we provide access to without introducing changes into the platform. This is one of the key aspects of the flexibility provided by the architecture.

To illustrate the above, consider the procedure for registering adapters. This procedure involves the *adapter provider* (the one that hosts the adapter) registering the adapter definition using the service provided by the RSMS' access layer for that purpose. This definition involves the mapping between the existing *resource types* (e.g., documents, pictures, etc) and *action types* (e.g., share, export, update, etc.) and the implementations provided by the adapter (and offered by the correspondent resource manager). This definition is then processed by the resource space manager, which registers these implementations. This is possible since resources types and action types have unique identifiers that allow reusing their definitions. However, nothing prevents an adapter to register new resource types and action types. In this case, these new definitions become available to other potential implementations.

As a result of the registration procedure, a new *resource manager* becomes available to the platform, implementing a set of actions and offering support for resources, sharing common functionalities with other *resource managers* semantically equivalent at given abstraction level.

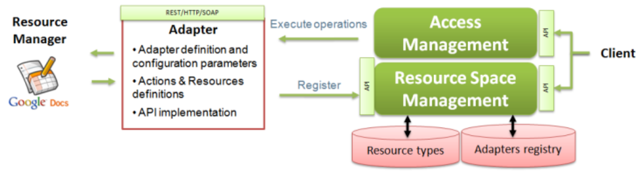

**Figure 7 Adapters registration and operations execution**

To make this more concrete, assume we want create a new resource type to operate, with the same actions, on documents subject to version control. Using ResMan, we have to perform the following call to the REST API:

```
POST http://project.liquidpub.org/resman/resource-type.xml
Body:
<resourcetype>
<name>Versioned Document</name>
<description> Resource type for versioned documents </description>
<user-ref>http://project.liquidpub.org/gelee/api/user/8901</user-ref>
<creation-date>2009-12-02</creation-date>
<actiontype-list>
<link href=http://project.liquidpub.org/resman/action-type/145
value="Ckeckout"/>
<link href=http://project.liquidpub.org/resman/action-type/141
value="Commit"/>
<link href=http://project.liquidpub.org/resman/action-type/144
value="Rollback"/>
  …
</actiontype-list>
</resourcetype>
ResponseLocation:
http://project.liquidpub.org/resman/resource-type/1.xml
```

The above call returns the URI to the newly created resource type. In the definition, we reference the action types that will be allowed by all the "Versioned Documents". Then, clients can get the resource type definition by asking ResMan about the resource type identified by the URI.

```
GET http://project.liquidpub.org/resman/resource-type/1.xml
Response:
<resourcetype>
    <name>Versioned Document</name>
…
    <adapter-list>
    <link href="http://demo.liquidpub.org/resman/adapter/38"
    value="SVN Client"/>
    </adapter-list>
</resourcetype>
```

In the RSMS extensibility approach, the resource manager and the concept of resource type collectively support a flexible binding approach that can range from static to dynamic binding to both adapters and (for services using the RSMS) to resources. Static binding to adapters is implemented by restricting (for a given RSMS client, or for all clients) access to a given (set of) resources to go through a specified adapter - and therefore using a specific mapping between generic actions at the resource type level and actual operations.

However in general it is possible to change dynamically the adapter we use to access a given resource: the mappings are specified and the adapters are registered, this is transparent to RSMS' access layer clients. Besides load balancing, the key benefit here is reliability and the ability to leverage the community to maintain a complex distributed system: in fact, sources, especially sources that do not assume they are accessed programmatically such as google scholar, change their interface from time to time and the parser/crawler needs to be changed accordingly. It is therefore possible that from time to time adapters became obsolete and returns errors. In this case the RSMS' access layer can dynamically switch to another adapter, and by keeping track of the last working adapter can also direct the choice towards one that has already embraced and implemented the change.

Notice that unlike traditional web service scenario, dynamic binding here is "provider-enabled" in that the provider of the adapter makes sure to define the mapping with the resource type actions as opposed to the RSMS (the "client" of the adapter "service") having to somehow figure out how to talk to the service or having to impose a standardization on the adapter interface.

## 6. USE CASE: LIQUID JOURNALS

As stated in the introduction, the Web has changed the way we create, consume, share and disseminate scientific knowledge. In this scenario, the obstacle to dissemination is not longer publishing, which can be achieved by simply putting a contribution online, but rather making a contribution visible (on the author's side) and quickly identifying interesting contributions in a sea of publications (from the reader's side). Yet, the current dissemination model continues unaware of these changes and obstacles, and so in the Web remains hidden a vast amount of interesting scientific content and new opportunities for creating, sharing, evaluating and disseminating knowledge, unexploited.

Through liquid journals, researchers can find and share "interesting" scientific content, such as blogs, experiments, datasets, "related" to a certain area of research. Interesting content is brought to the user usually by *querying the Web* for contributions matching her explicit and implicit preferences. These preferences go beyond the selection process and cover the evaluation, review and publication phases; and so, liquid journals support a whole spectrum of models from the more traditional ones to the ones more social and web-aware. This is mainly due to the deconstructed nature [9] of liquid journals that allows us to see the different roles of publishers as independent *services* provided by potentially different actors on the Web. Liquid journals[22] therefore represent an approach that leverages the opportunities and the lessons learned from the social web.

Besides the strong conceptual requirements in terms of models of dissemination, publication, collaboration and sharing, that is, redefining the notion of journal, building the liquid journal model implies modeling the *Web as a source*. This has both conceptual and infrastructural implications. Thus, as the core part of the model resides in leveraging the features offered by the Web, dealing with the underlying nature and problems of accessing Web resources just falls outside the real value of liquid journals as a model, and so, this could become the reason for not taking such interesting model into practice.

Here is where the sRSMS comes into play, providing the abstraction of the Web as a homogeneous source that liquid journals can query as it were a single database, i.e., the abstraction of *scientific resource space*. On top of this abstraction, liquid journals can build a conceptual model based on a consistent view of scientific web resources, and so embracing the new types of scientific contributions the Web has made possible. Therefore, from a conceptualization point of view, liquid journals can focus on defining collaboration and behavior models, and other journal-related concepts, while letting the sRSMS take care of the specifics.

More importantly, from the infrastructure point of view, the sRSMS provides the machinery for solving the heterogeneity of the underlying sources and mashing them up into uniform set of APIs for manipulating and querying the scientific web resources. Again, building the liquid journal infrastructure on top will concentrate the efforts on the high-level and actual journal features, such as capturing user interests and ranking the results according to their relevance.

Note that being part of the ecosystem built on top of the sRSMS, sharing the same underlying notions, will trigger high-level interactions and synergies. For example, services providing evaluation

[22] http://project.liquidpub.org/research-areas/liquid-journal

metrics can be beneficiated of the data of liquid journals and liquid journals can be beneficiated by these metrics, which could be used, for example, in the ranking. This is case for liquid journals and the *Reseval*[23] tool (see overall architecture in Figure 1). This synergy is encouraged by the resource space and mediated by the sRSMS. Recall the architecture, services on top can use and feed the resource space.

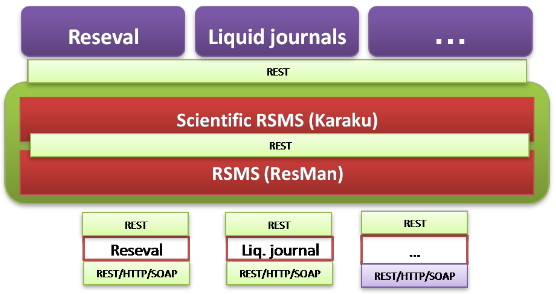

**Figure 8. Applications on top of the sRSMS**

Let us illustrate the interaction between the liquid journals application and the sRSMS by showing an example of how the sRSMS enables liquid journals to query the Web. Consider the case an author wants to get interesting contributions on the topic "Web services", and so she defines a liquid journal expressing this preference. Instead of limiting the contributions brought to the user to what is already on the system (as in social bookmarking services), the sRSMS enables the journal to go directly to the Web to get the contributions. This certainly makes the difference to the author. In Figure 9 we provide an example of how the user's ideal journal is translated into a query.

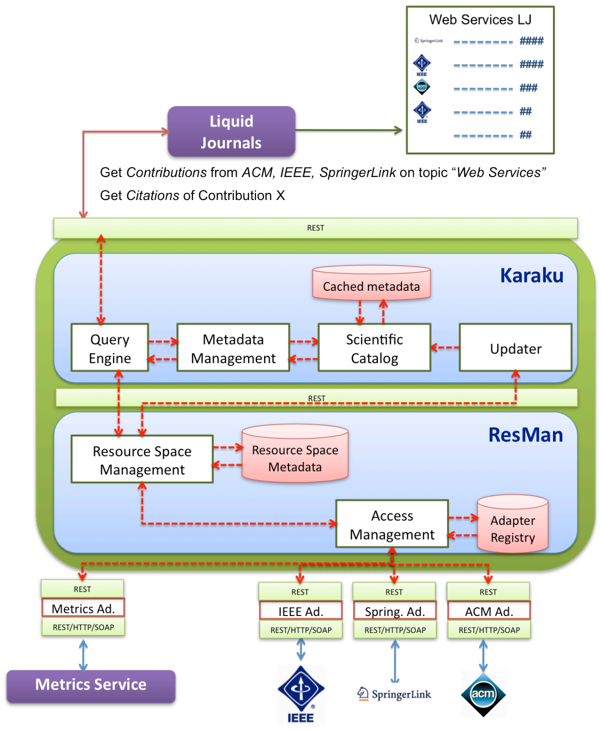

**Figure 9. Liquid Journal Use Case example**

[23] http://project.liquidpub.org/reseval/

As seen in Figure 9, executing this query is not trivial. The sRSMS needs to decompose the query expressed in terms of the scientific resource space entities, identify the adapters providing support for the resource managers selected by the user, translate the query to each adapter in terms of resources, and finally get the results back and join them according to the scientific resource space schema.

We can also see the workflow such query will follow. The process starts in the *query engine,* whose main job is to build the proper calls for the access layer based on the input query. Within the sRSMS, some metadata can be cached in the *scientific catalog* to answer queries faster. The *query engine* will also access this catalog and then pack all the results before deliver them to the client. The *scientific catalog* will be constantly updated by de *updater,* where some crawling and monitoring process are always running.

Once the query is parsed and expressed in the terms of resources (e.g., pdf files) and actions (e.g., search), the *resource space management* component will map them to proper *resource managers* (e.g., IEEE, ACM, SpringerLink, etc.). Given this, the *access management* component will use the resource managers' definition to find the corresponding *adapters*. The adapters then, will perform the calls to the actual service providers interfaces, getting the required resources to build the requested result.

At the end of the process, the Liquid Journals service will push all the results to the person's home page, enabling him to choose on a much more easy manner. We could go further and also add a connection to some metrics service (e.g., to get citation counts) to assess the contributions on the query result, providing more relevant information to support the decisions of the LJ editor.

Thanks to the extensibility properties of our sRSMS, all we need to enrich our LJ with a citation-based ranking is the corresponding adapter for calling the metrics service in order to get the "citations" resource.

## 7. CONCLUSION

In this paper, we have introduced concepts, an architecture, and an implementation of a Scientific Resource Space Management System (sRSMS). The system aims at providing a homogeneous view over and access to a space of scientific resources, in which the resources are sourced from the Web and accessible via a variety of different, heterogeneous technologies. Technological details are hidden to the users of the sRSMS via two layers of abstraction: first, we describe individual resources via resources types, and then we bind resource types to domain concepts. The final goal is to enable the users of the sRSMS to operate on the scientific resource space via domain-specific, intuitive instruments, such as the one represented by the Liquid Journal use case.

The innovative aspects of the proposed sRSMS are a combination of *universality*, which allows us to manage any web-accessible resource; *accessibility*, in terms of homogeneous and source-independent access to resources; *simplicity*, in terms of the general model and of the abstractions used, and *extensibility*, which is a property of both the model (which allows us to define different new resources and actions at different levels of abstraction) and of the architecture (that allows us to plug in new resource managers without introducing changes to the system).

The concepts, models and architectures are not theoretical only, but have been implemented in a functional prototype of as RSMS. The code is available in open source and we invite the reader to contribute to these and other tools of Liquidpub. Our future works include integrating the sRSMS into the Liquidpub platform, extending the resource space to other related domains, and analyzing new usage scenarios to improve the sRSMS's applicability.

## 8. REFERENCES

[1] M. Franklin, A. Halevy, D. Maier. From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.* 34, 4 (Dec. 2005), pp. 27-33.

[2] A. Halevy, M. Franklin, D. Maier. Principles of dataspace systems. *PODS*, 2006.

[3] D. Braga, S. Ceri, F. Daniel, D. Martinenghi. Optimization of Multi-Domain Queries on the Web. *VLDB 2008*, pp. 562-573, Auckland, New Zealand, August 2008

[4] B. Benatallah, F. Casati, F. Toumani. Web services conversation modeling: A Corner-stone for EBusiness Automation. *IEEE Internet Computing*, 8(1), 2004.

[5] S. R. Ponnekanti and A. Fox. Interoperability among Independently Evolving Web Services. *Middleware '04*. Toronto, Canada, 2004.

[6] B. Benatallah, F. Casati, D. Grigori, H. R. M. Nezhad, and F. Toumani. Developing adapters for web services integration. *CAiSE*, 2005.

[7] Marcos Báez, Cristhian Parra, Fabio Casati, Maurizio Marchese, Florian Daniel, Kasia di Meo, Silvia Zobele, Carlo Menapace, Beatrice Valeri: Gelee: Cooperative Lifecycle Management for (Composite) Artifacts. IC-SOC/ServiceWave 2009: 645-646

[8] H. Overdick. The resource-oriented architecture. *IEEE Congress on Services* (Services 2007), 2007, pp. 340–347.

[9] J. Smith, "Free Content The deconstructed journal–a new model for academic publishing," Learned publishing, vol. 12, 1999, pp. 79–91.