A Scientific Resource Space Management System

Cristhian Parra[1], Marcos Baez[1], Florian Daniel[1], Fabio Casati[1], Maurizio Marchese[1], and Luca Cernuzzi[2]

[1] University of Trento, Dipartimento di Ingegneria e Scienza dell'Informazione
Via Sommarive 14, 38123 Povo (TN), Italy
{parra,baez,daniel,casati,marchese}@disi.unitn.it

[2] 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 a 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 (sRSMS)*, since they are the resource analogous to data space management systems known in literature.

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) to specialized search engines (such as GoogleScholar) and metadata services (such as DBLP). In addition to these rather "traditional" means of knowledge dissemination, today's Web 2.0 is characterized by instruments that enable the *early sharing* of knowledge (such as wikis, blogs or personal web sites). These kinds of contributions are not peer-reviewed, but they might still have a huge impact on the scientific community depending on the reputation of their authors (think, for instance, of the so-called *technology evangelists*). Furthermore, there is an increasing interest in online repositories where scientists can publish, share and discuss their contributions, emulating the power and typical features

of *social applications*. Scientists might collaboratively enhance teaching material like slides or books and even share their data and experiments (e.g., myExperiment.org).

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 as a complement of 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 certainly 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 for the definition of standards (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 programmatically to interact with them. Unfortunately, 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.

Enabling users to search these types of scientific contributions therefore requires a novel approach, especially as for what regards the logic of *how to access* individual sources (multiple technologies might be involved in a single query) and of *how to abstract* them to the user. The goal of this paper is to extend the reach of services, such as those supported by traditional digital libraries, beyond their typical boundaries. To do so, we leverage some ideas taken from dataspace management [1] and 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; (ii) abstracting the *operations* that the services support, providing access to scientific knowledge (from simply accessing paper data/metadata, to extracting and tagging content, crawling citations, 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, Blogspot, or wikis).

Motivating scenario. The idea of sRSMS was born in the EU project *Liquidpub*[1], 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

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

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.

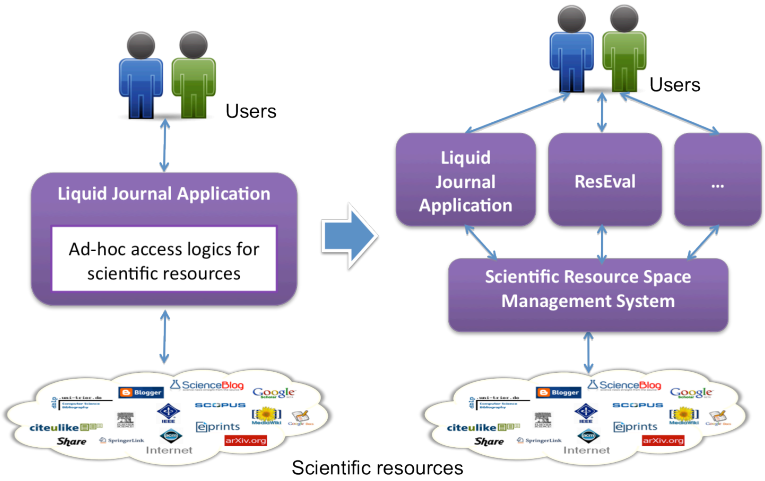

Figure 1. From ad-hoc access of scientific resources to a dedicated Scientific Resource Space Management System

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 reused in practice? What does our resource space look like? How do we deal with the heterogeneity of resources? How do we query the resource space? And so on.

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

- We introduce the idea of **Resource Space Management System** (RSMS) and its **scientific** domain counterpart (sRSMS), by describing 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.
- We provide some minor details about the **implementation of our sRSMS**, which is able to provide homogenous programmatic access to resources and web services, regardless of how they are implemented.
- We show how our sRSMS can be **leveraged to ease the development** of the Liquid Journal Application described above.

- Finally, in doing so, we aim at **simplicity, flexibility and collaborative extensibility**. Our sRSMS facilitates extensibility by allowing the community of developers to just register services that interface with systems or scientific resources available in the web and that may be hosted also by other parties (there is no need for plugging code in). Further details about this work can be found in the technical report version available online [10].

Next, we discuss some works that are related to the idea of sRSMS. In Section 3, we derive a set of general requirements for resource space management, which we then use in Section 4 to define our idea of scientific resource space management system. In Section 5 we show how such can be applied in the context of the Liquid Journal case study, and then we conclude the paper.

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]. We extend the principles of dataspaces to a *space of scientific resources*, where resources also have their 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 not only focus on the search problem, but we also provide abstractions for operating with scientific entities.

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. 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. The problem is that general-purpose search engines lack the necessary domain concepts and interaction capabilities to properly handle scientific resources.

The Search Computing (SeCo) project aims at answering complex queries by automatically deriving from the query input a suitable query execution plan, which can be used to orchestrate the interaction with individual search services [3]. The goal SeCo 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 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 as 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.

Our research also considers the problem of operating on sources. 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 this paper build on [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, Web of Knowledge, CiteSeer, DBLP, or GoogleScholar, which typically offer a much better and more flexible access to their content. They can answer queries that are more complex than simple keyword queries. 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.

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 this line, some relevant works to follow are the *Dublin Core Metadata Initiative* (DCMI), the *Open Archives Initiative Protocol for Metadata Harvesting* (OAI-PMH), the *Online Information Exchange* (ONIX) and the *Digital Object Identifier* (DOI®). All these 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.

3 Resource Space Management: concepts and requirements

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.

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.

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 func-

tionalities that justify the development of a dedicated RSMS. 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 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 as basic *services* (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 defines the actual resource space. 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 necessary metadata 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.

4 Managing and Accessing Scientific Resources

A generic RSMS as discussed in section 3 allows us to interpret the Web as a resource space in which all URI-accessible artifacts are resources. In this paper, the aim is to go beyond the mere technology abstraction and provide also suitable domain concepts that simplifies the access of and interaction with resources, and represents them in a way that can be understood by non-IT users 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*[2]. Then we describe how we tackle the problem of accessing the actual resources by the means of our base architectural layer called *ResMan*[3] [7].

4.1 The Scientific Resource Space (sRSMS)

In the context of scientific knowledge, a *scientific resource* is any resource that represents an important concept in the domain of scientific knowledge dissemination. Examples are documents (e.g., a Google Doc or a blog entry), experiments results and their datasets, metadata information like DBLP's records, authors themselves and so on. A *scientific resource space* contains all these scientific resources.

To support and push forward a group of innovative scientific services, first we need to speak the same language used in the domain of scientific research. The first step consists then in defining a comprehensive conceptual model that supports all possible entities and relationships of the specific domain and that will be common for all services built upon this layer. Many initiatives are being done to come up with such a model (e.g. OAI-PMH) but none of them have had enough impact as to become the industry "de facto" standard. We therefore introduce our own model, which is tailored to the specific features we want to support in the sRSMS.

Karaku Conceptual Model. The scientific resource space is modeled by means of three basic constructs (very similar to the well-known Entity-Relationship notation):

---

[2] 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.

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

- **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** (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** represent extended information attached to both relations and entities (e.g., comments, specialized attributes like tags). Annotations can be used to improve search performance and to specify how to bind entities to actual concepts.

The former 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 the system to manage resources more easily while also providing guidelines for further specializations.

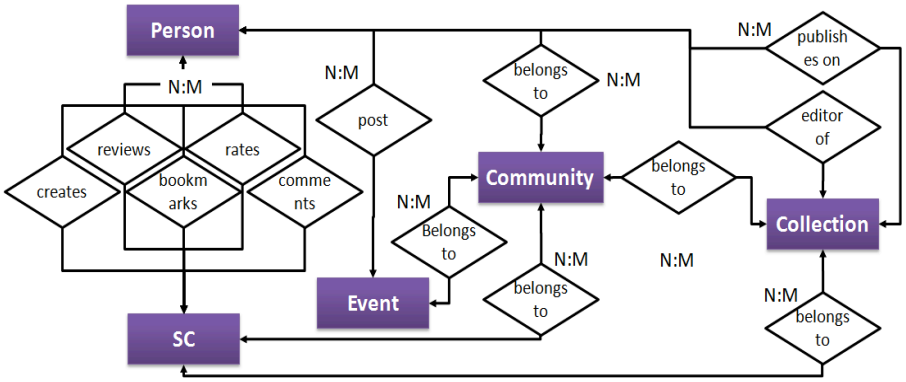

Figure 2. Conceptual model of the scientific research space

Figure 2 shows the model of the scientific resource space supported by Karaku. 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**: refer to groups of people working in a same field or area of research. Knowledge about communities and the involvement of people in them is particularly interesting to assess the "quality" or impact of a researcher within his community. Communities typically evolve frequently over time.

iv) **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), bringing together people for discussion and publication of scientific results.

The essence of this model 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 introduced modeling constructs), offering in this way an opportunity to explore the concepts that form the foundations of scientific activity.

Furthermore, 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.

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.

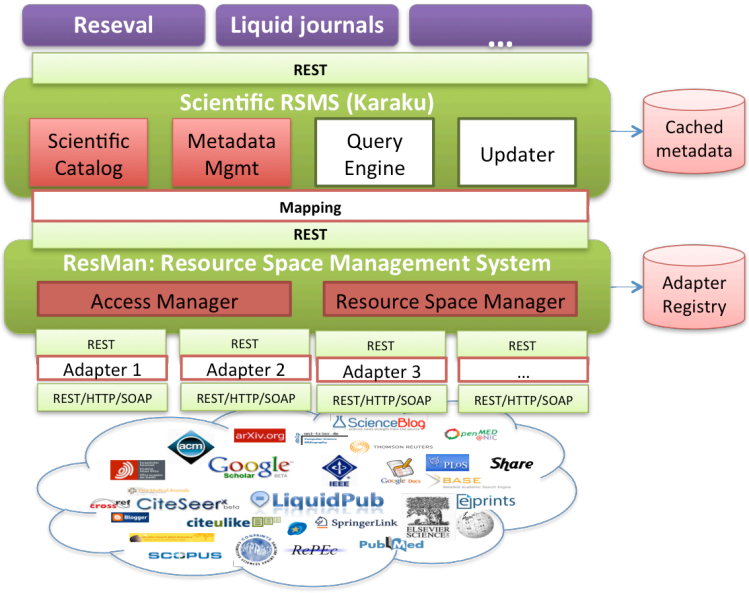

Fig. 3. Architecture of the two-layered sRSMS

ii) **Query Engine**: provides the mechanisms to answer the queries of the clients, expressed in a domain-specific query language expressed over the scientific cata-

log. 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 Create, Read, Update, Delete (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. In the whole project's design and implementation we have followed a *resource oriented architecture* approach [8] to be compliant with the latest trends 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 model.

4.3 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. Some examples of currently available and relevant scientific domain services are DBLP, GoogleScholar, SCOPUS, CiteUlike. All of them provide access to their content (the scientific contributions) via different APIs/protocols (e.g. own APIs, OAI-PMH, periodically updated datasets, etc.).

The main function of our access layer architecture *ResMan* (See Figure 3) is to abstract the technical *access-to-resource* specifics, providing for them a universal resource space access layer. In this section, we present only a brief discussion of this layer and we refer the reader to the Technical Report [10] for further details.

The Basic Resource Space. Figure 4 shows the conceptual model that introduces the necessary concepts and that is also the relational model for the *resource space catalog* of ResMan.

The first two elements of our model, *Resources* and *Resource Manager,* have already been introduced before. 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 allows us to operate with the resources. The basic elements provide operations and properties, which are specific to

the actual resource managers. Each of these elements however, can be naturally abstracted to support arbitrary resources at different levels of abstractions using a homogeneous interface.

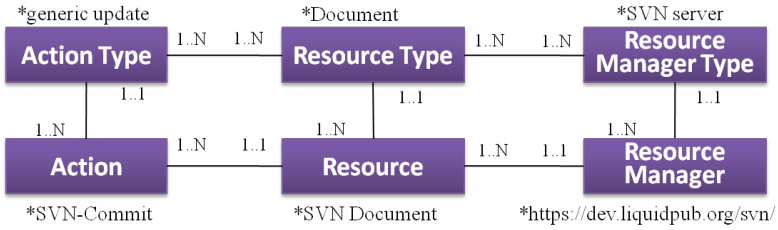

Figure 4. Resource space conceptual model

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. We include in our model the *action type* as a way of providing a common interface for these semantically equivalent actions.

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 3, 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.

The *resource space management* module provides the means for exposing 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 *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.

The interaction with the resource managers (the services providers) is performed through *adapters* [6]. 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 Use Case: Liquid Journals

LiquidJournals[4] is one of the services within the LiquidPub framework that aims to improve the way scientific knowledge is shared and disseminated. The deconstructed nature [8] of liquid journals allows us to see the different roles of publishers as independent *services* provided by potentially different actors on the Web. It 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*.

Here is where the sRSMS comes into play, providing the abstraction of the Web as a homogeneous source that liquid journals can query as if it were a single database. Let us illustrate the interaction between the liquid journals application and the sRSMS by showing an example. Consider the case an author wants to get interesting contributions on the topic "Web services", and so 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 5 we provide an example of how the user's ideal journal is translated into a query.

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 and merge 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*

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

catalog to answer queries faster. The *query engine* will also access this catalog and then pack all the results before delivering them to the client. The *updater*, where some crawling and monitoring processes are always running, will constantly update the *scientific catalog*. 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.). The *access management* will then use the resource managers' definition to find the corresponding *adapters* that will perform the calls to the actual service providers, getting the required resources to build the requested result.

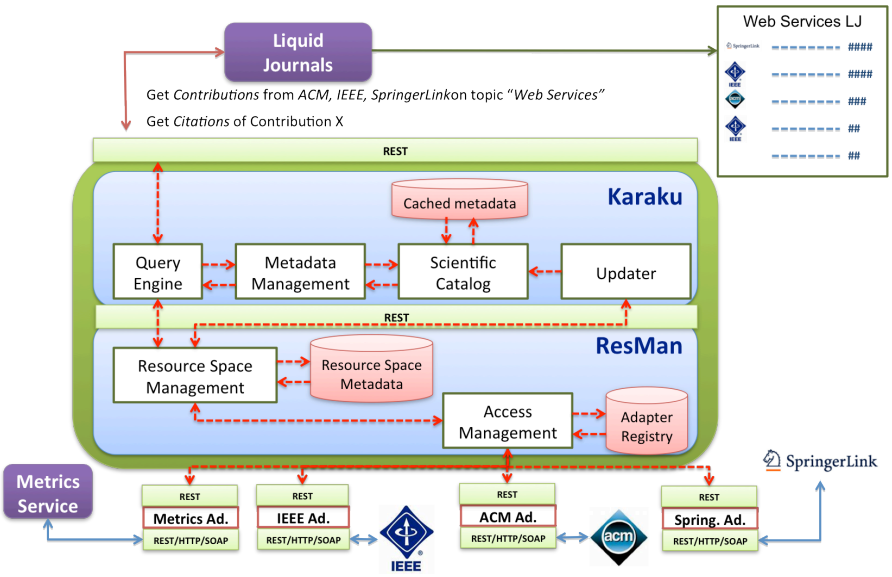

Figure 5. Liquid Journal Use Case example

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 more easily. 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.

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 goal is to enable the users of the sRSMS to operate the scientific resource space via domain-specific, intuitive instruments, such as the one shown in 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 an RSMS. The code is available as 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.

Acknowledgement. This work has been supported by the EU ICT project LiquidPublication. The LIQUIDPUB project acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 213360.

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 Cornerstone 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. M. Baez, C. Parra, F. Casati, M. Marchese, F. Daniel, K. di Meo, S. Zobele, C. Menapace, B. Valeri: Gelee: Cooperative Lifecycle Management for (Composite) Artifacts. ICSOC/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.
10. C. Parra, M. Baez, F. Daniel, F. Casati, M. Marchese and L. Cernuzzi. A Scientific Resource Space Management System. Technical Report DISI-10-013, Ingegneria e Scienza dell'Informazione, University of Trento. 2010. http://eprints.biblio.unitn.it/archiveb/00001812/