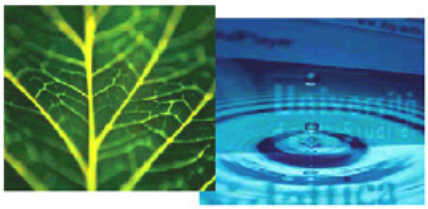
PhD Dissertation



International Doctorate School in Information and Communication Technologies

DIT - University of Trento

Knowledge Dissemination in the Web Era

Marcos Baez

Advisor:

Prof. Fabio Casati

Università degli Studi di Trento

November 2012

Abstract

In this research work we address the limitations of the current scientific knowledge dissemination model to face the new scenario posed by the Web. We explore the historical reasons behind the current model, and we show that it is essentially the same today, even if the Web has made dissemination nearly real time and free. We show how this misalignment between the current model and Web capabilities brings not only missed opportunities but also a serious overload problem, creating difficulties for authors to gain visibility, and for readers to find interesting content. Our approach has been to build from the ground up, by i) understanding how the very core concepts of the scientific publishing such as "scientific contribution", "scientific journal" and even "reputation" should adapt to the new scenario; ii) studying the dissemination and sharing practices that are inherently present in the scientific community, to understand how technology can empower those practices to reduce information overload. The outcome is novel models, tools and a platform for knowledge dissemination that takes full advantage of the Web while addressing the overload problem.

Keywords
information overload, knowledge dissemination, scientific publications

Contents

1 Executive Summary **1**

1.1 The Problem 3

1.2 Our Approach to Knowledge Dissemination 5

1.3 Structure of the Thesis 9

2 Addressing Information Overload in the Scientific Community **13**

2.1 Introduction 13

2.2 The Liquid journal model 16

2.3 Usage model and metrics 22

2.4 Architecture 24

2.4.1 Related work 26

2.4.2 Conclusion 28

3 Understanding and supporting search for scholarly knowledge **33**

3.1 Introduction 34

3.2 Understanding search 36

3.2.1 Qualitative Analysis Results 37

3.2.2 Quantitative Analysis Results 41

3.3 Defining Network-aware recommendations 43

3.3.1 Formal problem definition 43

3.3.2 Defining the notion of network 44
3.3.3 Defining popularity functions 46
3.4 Evaluation 48
3.4.1 Experiment definition 48
3.4.2 Dataset Description 49
3.4.3 Running and analyzing the experiments 49
3.4.4 Discussion 52
3.5 Related Work 53
3.6 Concluding remarks 55

4 Sharing Scientific Knowledge with Knowledge Spaces 57
4.1 Introduction 58
4.2 Knowledge Spaces 62
4.2.1 Instant Communities scenario 62
4.2.2 Scientific Resource Spaces 64
4.2.3 Kspaces Conceptual Model 65
4.3 Knowledge space platform and services 67
4.3.1 Architecture 67
4.3.2 Collecting and posting knowledge 69
4.3.3 KS Intentional Language 70
4.4 Knowledge space Applications 73
4.5 Related Work 76
4.6 Findings, Status and Next Steps 80

5 Resource Space Management Systems 83
5.1 Introduction 84
5.2 Implications for Research Spaces Management Systems . . 86
5.3 Scientific Resource Space Management 88
5.4 Generic Resource Space Management 89
5.4.1 Resource Space Model 90

5.4.2 Architecture and services 92
5.4.3 The role of adapters 94
5.5 Use Case: Liquid Journals 97
5.6 Conclusion 102

6 Universal Resource Lifecycle Management 105
6.1 Introduction 106
6.2 Motivating Scenario 109
6.2.1 EU Projects 109
6.2.2 Problem and Requirements Abstraction 111
6.3 Related Work 113
6.3.1 Workflow Management Systems 113
6.3.2 Document Management 114
6.3.3 Lifecycle Modeling Notations 115
6.4 Concepts, Models and Languages 116
6.4.1 Lifecycle model: basics 117
6.4.2 Lifecycle Execution 119
6.4.3 Actions 121
6.4.4 Roles and Access Rights 122
6.5 Gelee at Work 123
6.5.1 Overall Architecture 123
6.5.2 Resources and Actions 124
6.6 Conclusion and Future Work 126

7 Lessons Learned 129
7.1 Impact on knowledge dissemination 129
7.1.1 Liquid journals 130
7.1.2 Instant Communities 130
7.1.3 Knowledge Spaces Platform 134
7.2 Final Remarks 135

Bibliography **136**

List of Figures

2.1 Example of scientific resources connected by next-version-of relation 16
2.2 Example of different representation for the same resource . 17
2.3 Example of other general relations 17
2.4 LJ backend architecture 30
2.5 LJ frontend collage 31

3.1 Sources of references in a research paper, classified by seniority of the researcher 38
3.2 Social networks of origin for references 40
3.3 Results of our online survey, classified by seniority of the researcher 42
3.4 Social references by social network 43
3.5 Precision of the different popularity functions by year . . . 50
3.6 Recall of the different popularity functions by year 51

4.1 Kspaces and KS designs are ways to create, share and consume resources. 65
4.2 Architecture of the KS platform. 68
4.3 Example of Interface using KS intensional language. 72
4.4 Instant Community Application UI. 77

5.1 Architecture of the Karaku sRSMS 89
5.2 Resource Space model 91

5.3 Scientific Resource Space Architecture 92
5.4 Adapter registration and operation execution 94
5.5 Adapter registration and operation execution 100

6.1 EU Project deliverable lifecycle. 118
6.2 Gelee high-level architecture. 124

7.1 Liquid Journals sreenshots 131
7.2 Liquid Journals on MIT Techonology Review and BMJ (British Medical Journal) Blog, and other publishing and library blogs 132
7.3 Instant Community at various events 133

Chapter 1

Executive Summary

The scenario in which current scientific research is undertaken has greatly changed since the days printed journals, letters, and conference talks were the only form of scientific knowledge dissemination. In those days, the scarce and expensive resource was the *printing* and *distribution* of *papers*. Publishing was expensive. As a result, there was a need to screen contributions before they got published, and there were no other means to do this than peer review. Besides the unavoidable time delays, the process was the only reasonable one, and it was feasible as the research community was relatively small and the reviewing effort was low. Printing and distribution also means, for example, that journals had to be organized in volumes and *issues*, available periodically.

The Web has changed the way we create, consume, share and disseminate scientific knowledge. Publishing is now almost real time and free and papers are not longer the only form of scientific dissemination. We can now publish early ideas in blogs (e.g., science blogs, blogger), put pre-prints in online repositories (e.g. arXiv, eprints), experiments, datasets and slides on our homepage. A brand new world of possibilities is opened for how scientific knowledge dissemination, creation and evaluation could be done and for how the notion of *scientific contribution* could evolve to serve the need

of scientists to learn about novel, interesting research ideas and results.

What the above implies is that the original reasons for having the current dissemination model (content given by papers, journals divided in issues, submission and peer review process) are gone. It does not mean, however, that the dissemination model should not be based on properly written papers and peer-review as prime process. The point we make is that before the web these were the only options, now that we are freed of the constraints of printed materials, a world of new possibilities arises.

But we are facing not only opportunities in this new scenario. These changes, along with a developing educational and economic environment that has allowed an increase in the number of people doing research, have caused a tremendous increase of knowledge artifacts that are disseminated every day. This means that the scarce resource is now attention [26] and the obstacle to dissemination and the challenge for scientists - is not publishing but rather making a contribution visible (on the authors side) and quickly identifying interesting contributions (from the reader s side).

In this research project we endeavored to study whether this model is efficient and effective, which alternative models can support scientific dissemination, and how IT can support them.

Indeed, studies shed many doubts on the validity of this approach and certainly question whether this can be the only or even the preferred model for scientific dissemination. There is no scientific evidence that peer review is effective in selecting high impact papers, while there is scientific evidence that citations are not correlated with impact (e.g., [22] [31]). But one of the most important finding is that people do not find scientific knowledge or ideas by searching on search engines. At most they can retrieve the PDF from there.

On the contrary, scientists tend to stumble upon knowledge [9]. It can happen when a colleague points us to a paper, when we listen to a

presentation at a workshop, or when somebody makes a connection we did not think of, or mentions an idea at the water-cooler. These are the main ways in which we interact. What this tells us is that the most useful forms of interaction are not based on the paper as found by searching the web or reading a journal. Instead, they can happen in any form (a paper, but also a presentation, a video, a comment, or a "link" among them that makes us see a connection). It also tells us that exchanges happen socially.

Given this, what we endeavored to do is to enable these kinds of exchanges, or, to put it differently, to help people stumble on knowledge in all its forms, and to help create conversations on this knowledge. And the Web is the key ingredient for this. The results of the research we have done clearly hints at the fact that a key role for future web applications is indeed that of helping us capture, share, and find scientific knowledge, and as such enables scientific conversations that help us create knowledge efficiently and sparkle new ideas.

1.1 The Problem

Thus, we live in an era in which we are exposed to an almost infinite ammount of scientific knowledge of various kind, and yet we use essentially the same organisation, models and processes as in the pre-web era. This discrepancy brings some noticeable consequences and limitations to scientific knowledge dissemination:

- **Information overload**. It refers to the problem of scarce attention that makes it difficult i) for readers to find interesting and relevant scientific contributions and ii) for authors to make their contributions visible. This is perhaps the most noticeable problem to which the others below are closely related.

- **Outdated notion of scientific contributions**. The current model of "scientific contribution" is basically a digital representation of the printed paper. There are many other types of contributions being produced during research such as datasets, experiments, simulations, slides, or even ideas that deserve as much attention as (if not more than) the traditional paper. Currently these interesting types of contributions are either not accessible, hard to find or relate. Indeed, citations remain as the only way of relating scientific contributions even when the semantics of the relations might different and could serve different purposes. As a consequence the current model is not only missing opportunities but also contributing to the information overload problem.

- **Uneffective organisation**. Scientific contributions have traditionally been organised in proceedings, issues, volumes, using some topical categories. However, with the explosion in the number of scientific contribution these are not longer able to meet the more specific information needs and seeking behavior of the community [4].

- **Slower processes**. The processes involved in the current dissemination model (e.g., peer review, publishing) are comparatively slower than the real-time nature of communications and publishing on the Web. This is a problem particularly for authors who need to wait for months to get their work published or even just feedback from their peers. Given the pace at which research is pushed nowadays, there is a need to understand how the Web, and specially the social Web, could help in bringing the scientific dissemination much closer to the speed of current communications.

- **Outdated research activity rewarding**. Evaluation is a necessary aspect of research, and as such it should encourage not only one

particular type of research activity (e.g., writing papers) but also behaviours that are good for science. Current metrics reward only paper writing (using mostly citations), not contributing much to reducing the problem of information overload.

In the recent years, some progress has been done in the dissemination model to address some of the issues above. An indication of this is the publication of papers (or preprints) in open/free repositories and archives; new journal models experimenting different review and publishing models (e.g., [49] [42]), and new metrics aiming at evaluating more fairly the work of researchers(e.g., [25]). Even though these initiatives did alleviate some of those problems, they are still based on the traditional notion of scientific contribution and were not designed to take full advantage of the Web and more importantly to deal with the information overload problem.

We have also seen an increasing number of tools and services allowing publishing, searching, sharing and bookmarking of scientific papers. Prominent examples are Google Scholar, Mendeley, ResearchGate and CiteSeer, which facilitate the process of finding and discovering scientific papers. However, these services do not consider the practices and strategies of the community when it comes to finding and sharing scientific information and rather provide separate solutions that, while technologically strong, do not fully exploit the specifics of the domain. This is particularly true in our community, where most of the sharing is done in different contexts (e.g., talks, conferences, after reviews), with different networks (e.g., colleagues, people at conferences) usually in very informal settings.

1.2 Our Approach to Knowledge Dissemination

In the light of these new opportunities and challenges, the goal of this thesis has been to identify models, develop tools and services for knowledge dis-

semination that, by design, would embrace the benefits of the Web while addressing the problems caused by the information overload. Thus, our approach has been to build from the ground up, by i) understanding how the very core concepts of the scientific publishing such as "scientific contribution", "scientific journal" and even "reputation" should adapt to the new scenario; ii) studying the dissemination and sharing practices that are inherently present in the scientific community, to understand how technology can empower those practices to reduce information overload; iii) facing the techonological challenges in designing models, services, tools and architectural support for the solution.

Analyzing the current dissemination model helped us to understand the weaknesses of the traditional models that needed to be addressed. Besides a fundamental discussion of what is a scientific contribution, we propose a model that is able to better capture the "multi-faceted", "evolving" and "connected" nature of scientific knowledge. We also proposed novel ways of organising and disseminating scientific knowledge, building on the traditional model of scientific journal. Our proposals are meant to address some core problems of the dissemination model that were contributing to the information overload problem.

Studying the information seeking and sharing practices led us to interesting findings. According to our results [9], scientists tend to stumble upon knowledge as opposed to searching. It can happen when a colleague points us to a paper, when we listen to a presentation at a workshop, or when somebody makes a connection we did not think of, or mentions an idea at the water-cooler. These results, along with our analysis of the limitations of the current model, shaped our research towards helping people stumble on knowledge in all its forms, and helping create conversations on this knowledge.

What we understood during these years of research is that doing this has

two sides. First, we need to elicit this knowledge, or, to put it differently, to lower the barriers to sharing. People often like to express opinions, but often this requires too much effort. For example, sometimes when we hear a talk we think of some related work, or we think of something we'd like to point out to the speaker, but we just dont do it. Its too much trouble to interrupt, and then there is the coffee break coming up. The opportunity is lost and you and the speaker might never catch up again. Or, think when you're back from a conference and want to point out interesting presentations or papers to your colleagues. There is a lot of interesting knowledge such as discussions, questions or related work, that is lost on the way or simply not shared. The other aspect we understood is that capturing knowledge is context-specific. There is no single metaphor or tool that can work in all circumstances. In some cases these ways to capture knowledge are done through novel organizational models, in other cases through software platforms, most often it is a bit of both. These observation share analogies with Information Foraging Theory [43] and were exploited in our approach.

In summary, the innovative aspects and outcome of this thesis are:

- **A model of scientific contribution** designed for the Web. This new model separates the nature of the contribution (e.g., paper, dataset, experiment) from the level of certification (e.g., peer reviewed or not) and degree of maturity (e.g., workshop or journal paper, expeirment v1 or v2), thus broadening the range of scientific contributions to the opportunities of the Web. Moreover, it enables new associative organization metaphors (e.g., line of research), to *face the information ovearload problem*, by providing semantic relations. Other benefits of the model can be also described from the perspective of new opportunities in terms of knowledge creation and evaluation.

- **A model and prototype of scientific journal** for the Web era, namely "liquid journals", designed to address the information overload problem. The underlying principles consist i) in leveraging the very same (large) community of scientist that creates the overload problem/opportunity to collaborate in filtering and prioritizing the information, ii) in enabling a dissemination and consumption model that naturally reduces the noise portion of the information overload right at the source, iii) in having a set of metrics that mitigate the overload and encourage "good behaviors" for science, such as early sharing and providing feedback, and iv) by computing scientific diversity and enforcing it when providing information.

- **A metaphor, a set of models and processes, and a social web platform**, namely "knowledge spaces" (kspaces for short), that help you capture, share and find scientific knowledge, in all of its forms. The principle behind kspaces is to allow knowledge dissemination in the scientific community to occur in a way similar to the way we share knowledge with our colleagues in informal settings. It is based on the dissemination and sharing practices of the scientific community.

- **Tools for supporting knowledge sharing in seminars, conferences and courses**. Besides the conceptual contributions, a tangible outcome can be seen in the set of tools built on the foundations of this thesis. "Instant Communities"[1] and "StreamScience"[2], have been continously evolving over the years, supporting several conferences, workshops, seminars and courses. Thus, from mockups to prototypes they have evolved into solid tools open to the public and used by real users.

[1] `http://ic.kspaces.net`
[2] `http://www.streamscience.org`

- **Resource Space Management Systems** as an abstraction layer for integrating scientific services. It is based on a common view of the space of scientific contributions, and allows operating, searching and managing scientific resources disseminated across multiple services on the Web.

- **Lifecycle management model and system for scientific artifacts**. The model called "Universal resource lifecycle management" was designed to support agile environements where flexibility at design time (to allow lifecycle models evolve) and execution time (to allow users to deviate from predefied actions) are required. In this research project, we have developed a prototype, namely "Gelee", that had supported (at a conceptual level) flexible scientific processes in liquid journals and kspaces.

1.3 Structure of the Thesis

This thesis is structured as a compilation of the various research publications on the topics of this thesis. Thus, each chapter has been previously reviewed by peers and represents per se a contribution to a target research community.

Chapter 2. Addressing Information Overload in the Scientific Community.
Baez, M. and Birukou, A. and Casati, F. and Marchese, M. [4].

In this chapter we present a dissemination model that extends the notion of scientific journal to address the problem of information overload in the scientific community. We focus on the issues related to having access to interesting scientific content, and to narrowing down the discovery process

only to known sources (venues, authors) when dealing with vasts amounts of information. The contributions introduced in this chapter are a set of models, concepts, methods, and a supporting platform to address the overload problem.

Chapter 3. Understanding and supporting search for scholarly knowledge

Baez, M. and Mirylenka, D. and Parra, C. [9]

In this chapter we moved to understanding the information seeking practices that are inherently present in the scientific community. We took as a particular case the problem of finding relevant references and run interviews with groups of researchers asking them how they address the problem. The results of this study suggest that finding scientific knowledge has a strong social component, with the different researchers' social networks (e.g., coauthors, people met at conferences) accounting for an important percentage of the source of the references. In this chapter we report on this study and compare our results with the evidence found in analyzing the citation network of a dataset of 5×106 authors with their publications and references. We take these results and analyze different approaches for incorporating the social component into search and recommendation of scientific publications.

Chapter 4. Sharing Scientific Knowledge with Knowledge Spaces

Baez, M. and Casati, F. and Marchese, M. [7]

This chapter presents a set of models and an extensible social web platform (namely, Knowledge spaces) that supports novel and agile social scientific dissemination processes. Knowledge spaces is based on a model for sci-

entific resources that allows the representation of scientific knowledge and meta-knowledge, of effective viral algorithms for helping scientists find the knowledge they need, and of interaction metaphors that facilitate its usage. The concept and a implementation of Knowledge spaces, in their various forms and designs, are being exploited in several different pilots in cooperation with IEEE, the EU Commission, Springer, the archeology museum in Cambridge and major international conferences to support the collection and sharing of knowledge in scientific communities.

Chapter 5. Resource Space Management Systems

Baez, M. and Casati, F. [5]

Parra, C. and Baez, M. and Daniel, F. and Casati, F. and Marchese, M. and Cernuzzi, L. [39]

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 chapter 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.

Chapter 6. Universal Resource Lifecycle Management

Baez, M. and Casati, F. and Marchese, M. [6]

This chapter presents a model and a tool that allows Web users to define, execute, and manage lifecycles for any artifact available on the Web. In the chapter, we motivate the need for lifecycle management of Web artifacts, and we show in particular why it is important that non-programmers are also able to do this. We then discuss why current models do not allow this, and we present a model and a system implementation that achieves lifecycle management for any URI-identifiable and accessible object. The most challenging parts of the work lie in the definition of a simple but universal model and system (and in particular in allowing universality and simplicity to coexist) and in the ability to hide from the lifecycle modeler the complexity intrinsic in having to access and manage a variety of resources, which differ in nature, in the operations that are allowed on them, and in the protocols and data formats required to access them.

In the following, we expand on the contributions of this thesis to knowledge dissemination in the Web era.

Chapter 2

Addressing Information Overload in the Scientific Community

Baez, M. and Birukou, A. and Casati, F. and Marchese, M.

In this paper we present a dissemination model that extends from the notion of scientific journal to overcome the problem of information overload in the scientific community. We focus on the issues related to having access to interesting scientific content, and to narrowing down the discovery process only to known sources (venues, authors) when dealing with vasts amounts of information. In this paper we present the liquid journal model, concepts, methods, and the supporting platform.

2.1 Introduction

The Web has opened a whole world of possibilities for how scientific knowledge can be created, evaluated and disseminated. We can now publish preprints in online archives (e.g., arXiv) or simply post our papers on Web

pages. Furthermore, "papers" are not the only unit of scientific dissemination. Data, comments, scientific experiments, and even blogs can now be shared and they can be considered a form of scientific contribution that can help other scientists in their work. This means that today we have a large scientific community who can make available a large, and rapidly evolving, set of scientific contributions of different kinds. An implication of this is that while in the past the scarce resource in scientific dissemination was printing, now it is attention. The obstacle to dissemination is now to be able to find interesting and relevant information (for readers) and to get the work visible in the sea of virtually infinite information (for authors).

An additional and somewhat puzzling problem of information overload is that with so much information available we would at least hope to be able to broaden our horizons. For example, we would hope to be able to search for contributions on "the effectiveness of peer review" in many different domains (as this problem is indeed studied in different areas). However, having this much information results in narrowing down what we read as opposed to broadening it. We experience this in everyday life: having a TiVo[1] or analogous digital video recorder makes a wide set of TV programs available to us but at the end we tend to watch/record what we know we like, and are less encouraged to look for new programs. The same effect has also been observed in science [47], as we tend to keep looking into the sources we are familiar with, thereby missing a plethora of potentially interesting and relevant contributions.

Today we have very few tools at our disposal to leverage the richness of information while handling the overload. When we search for contributions, we still tend to look for papers, and one option is to do so by looking at collection of papers indexed by services such as DBLP (essentially for titles) or Citeseer. This is useful but it does not come near to solving the

[1] `www.tivo.com`

problem: we are limited to what is indexed, we are limited to papers (and to published papers), we are narrow in the selection (e.g., these services are in computer science for the most part), and despite this narrowness we are still likely to be overloaded with the result. An alternative approach is to use a google search, but that is not tailored to finding scientific contributions. Or we can use Google Scholar, the specific search for scientific contributions offered by Google, but the result is not often as helpful as when we search the Web for other purposes. Furthermore, even when we find something we like, we can only "navigate" to related content via citations, inserted by the authors at the time of writing.

In this paper we propose the notion of Liquid Journals (LJ) as a way to overcome the information overload issue in scientific publications. Their underlying principles consist i) in leveraging the very same (large) community of scientist that creates the overload problem/opportunity to collaborate in filtering and prioritizing the information, ii) in enabling a dissemination and consumption model that naturally reduces the noise portion of the information overload right at the source, iii) in having a set of metrics that mitigate the overload and encourage "good behaviors" for science, such as early sharing and providing feedback, and iv) in facilitating readers in linking knowledge in order to support other users subsequent search and navigation through related content.

LJs put this principles at work through concepts, methods, and ultimately tools. In the following, we present the usage models and derived metrics. Finally we describe the architecture and the prototypal implementation of the model.

2.2 The Liquid journal model

The liquid journals model builds on a model for scientific contributions, which is designed to facilitate the search for - and navigation of - scientific information of interest. We see scientific contributions as a structured, evolving, and multi-facet objects. Specifically, we see the space of scientific content we want to search and help to assess and disseminate as consisting of scientific resources, organized as set of nodes in a graph, that can be connected and annotated by authors, editors or even readers. The reasons for connections, and hence for modeling resources as a graph, is to capture several kinds of dependencies or relationships among them (or between resources and people or other entities, as discussed next).

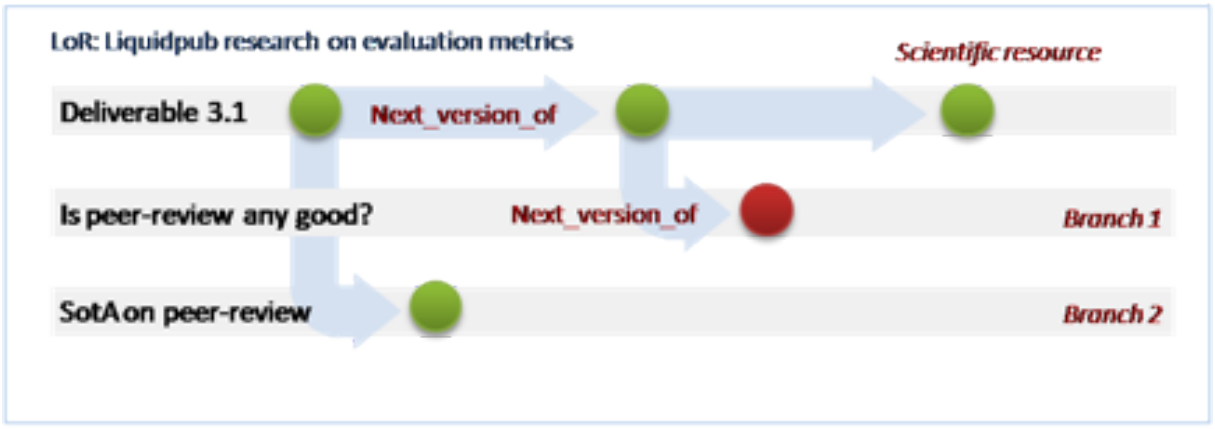


Figure 2.1: Example of scientific resources connected by next-version-of relation

To illustrate these concepts, in Figure 2.1 we show the work of our research group on evaluation metrics and peer review. We started this line of research within the context of a project deliverable (D3.1). This deliverable is composed of a review of the state of the art, experiments, analysis and presentation of the results. The results were delivered in two releases, D3.1v1[2] and D3.1v2[3] , and we plan to produce in the near future a third

[2] https://dev.liquidpub.org/svn/liquidpub/final/Year1/LP_D3.1.pdf
[3] https://dev.liquidpub.org/svn/liquidpub/wp3/d3.1/v2/

Figure 2.2: Example of different representation for the same resource

Figure 2.3: Example of other general relations

version. These releases are captured by special relations that allow us to specify that a particular scientific resource is the evolution, or a new version, of a previous one. Then, at some point, we reached some interesting result we wanted to communicate, so we took some of the work of the second version of our deliverable and produced a technical report: "Is peer review any good?". This type of spin-off is captured by different branches in the graph of the line of research. Then, when expanding a particular scientific resource (is peer review any good?), we can see that the very same paper has many representations (Figure 2.2). These alternative representations are the different views of the same resource, as in the example:

slidesets, a technical report and a conference paper. We can also see how this scientific resource is semantically related to other entities. In Figure 2.3 we illustrate the use of particular dataset and experiments (conference review data and code that processes them). These links allow us (or the editor) to connect and describe relationships among resources.

The reason for allowing anybody to define relationships is because in this way we can leverage the power of the community to build scientific dissemination knowledge, that is, knowledge that can help annotate and relate resources above and beyond what authors would do. In other words, people generate knowledge that helps in organizing and finding scientific resources. This is sometimes called "second-order knowledge", which we believe is as important in supporting scientists' work ("standing on the shoulders of giants") as first order one.

Formally, we define the space of scientific resources as $\Sigma =< SR, E, L, A >$ where

- SR is a set of resources $r =< id, uri, ct, cf >$ are the individual scientific resources. id denotes the universal identifier for the resource. uri points to the resource as available on the Web. ct is the content type of the resource and can take values such as paper, video, slideset, dataset, experiment, and others. cf is the content format which can for example be pdf, pptx, and so on. Because we consider journals as a way to create or at least to disseminate knowledge, they are also resources.

- E is the set of entities that create, access, relate, annotate, or certify resources. These can be people or institutions (including certification agencies).

- L denotes a set of links $l =< es, et, lt, u, un >$ representing relations among resources or between resources and entities (from source es

to target et). Besides the objects they relate, they are essentially characterized by a type lt (e.g., "next version of"), by the users $u \in E$ that created it, and by the set $un \in E$ of users or agencies that endorse it, if any.

- A denotes a set of annotations $a =< e, at, v >$ that can be attached to a resources or entity e. Annotations can be of a certain type at (e.g., tags, flags, comment), and carry a value v (e.g., "good example of state of the art").

This model for resources reduces overload because it clusters contributions and then allow users to navigate through contributions in the line and evolutions, presentations, and other related resources.

While the model allows anybody to create any kind of relation, the liquid journal model assumes and leverages specific relation types, to which it assigns an agreed semantic (and also graphical interaction patterns in the LiquidJournal interface).

- Structural relations represent arbitrary relationships between contributions, where the relationship is described by annotations. For example, a paper can be reporting on a dataset in that it describes results of experiments on that dataset. Examples of such relations are depicted in Figure 2.3.

- Temporal relations (such as next_version_of) model the evolution of a resource, be it a paper or dataset or anything else. This is a natural behavior of research dissemination where for example we write a preliminary version of a paper and then we extend or refine it. Or, we clean or add more data to a dataset. Figure 2.1 also shows that evolution can follow a line (as in multiple versions of our project deliverable 3.1) or branch (from one deliverable we then derive a paper or a technical report).

- Representation relations allow us to model the multi-faceted aspect. For example, a paper can have associated slides and datasets, and so be deemed as a complex multi-faceted artifact, including artifacts that encode (part of) the same knowledge but have different representation. An example of this type of relation is illustrated in Figure 2.2.

- Authorship relations denote who contributed to the creation of the resource. An annotation of this relation would qualify the contribution (e.g., "design of the experiment).

- Dissemination relations denote usage by mean of the LJ model. For example they include appearance of a resource in a journal, subscription to a journal, and sharing of a resource.

This model for resources reduces overload because it clusters contributions into research lines (which are themselves resources) and then allow users to navigate through contributions in the line and evolutions, presentations, and other related resources. Although outside the scope of this paper, it allows to more fairly attribute credit to contributions or authors by making explicit the incremental nature of a contribution and to assign indirect reputation to resources because they are linked by another (reputed) resource, much like what pagerank does for webpages.

A liquid journal is an evolving collection of interesting and relevant links to scientific contributions (whether freely or not) available on the web. Considering journals as collections of links means that journals do not own the contributions. We assume contributions are posted elsewhere[4] (web pages, traditional journals, etc) and so they are independent of their appearance in journal. Thus, many journals can then refer to the very same

[4]This means that availability of the actual data as well as access control and other aspects, cannot be ensured by the system. However, scientific contributions can point to reliable archives and, in general, sources that ensures the long term persistence.

contribution. This "appearance" of contributions in journals is an important information we exploit for measuring interestingness, as discussed below.

The links in a journal (which define its content) can be decided by the editor who picks them one by one, or can be defined by a web search through the liquid journal engine, where the results are dependent of the interestingness of the resources. The result of the search can be refined and then "snapshotted" by the editor (resulting in an issue of a liquid journal), or the journal can adopt a continuous model where the journal is essentially the web search, and the result evolves naturally and continuously as new content becomes available or as the values of metrics for existing contributions makes them qualified for the journal we have defined.

The rationale behind this model is that we see journals as a mechanism for people to find and share interesting and diverse content, for themselves or for their research group[5]. While doing this, while running LJ-enabled search for web content, and while refining the results and sharing the most interesting contributions with our colleagues, we do a service to our team but, as we will see, we are also acting as "filters" in that we implicitly rate contributions. Hence, we are also doing a service to the community. LJs essentially put the community itself to work as content selectors, while having people performing activities they need to do anyways, such as looking for content and sharing interesting findings with their team. It is like capturing the interestingness people perceive from the result of a web search, and using this as a way to rate content and therefore separate interesting contributions from the rest [30].

[5]This was also the original motivation at the birth of the scientific Journal paper model around the 17th century. It is also why we believe that our new model correctly mantains the name journal

2.3 Usage model and metrics

Liquid journals aim at providing tailored scientific content by bringing interesting scientific contributions. People fill their liquid journals in various ways: they can add content they stumble on (analogously to digging an item), by emailing a pdf file to the LJ engine, or even by taking a picture of a paper with their phone. They can also add a work-in-progress such as a google doc (see the demo videos for details[6]). This is intended to mimic what we do today to keep track of interesting contributions. The actual content, however, is not in the journal. A liquid journal is a collection of links, and as such, it relies on the actual sources and on the editor's ability to access sources. Thus, access permission are always based on the reader's permissions and on what the source of the link allows.

A value proposition of LJ is that editors and readers provide knowledge that can help connect and assess scientific contributions. This happens in 3 ways, all supported by the LJ interface:

1. Editors implicitly evaluate resources by publishing them in their journal.

2. Readers implicitly evaluate resources by sharing them with their team. For example a professor or a phd student may share paper they think interesting within their team.

3. Readers provide knowledge by linking and annotating resources. For example, a reader can state that paper P1 reports results of experiment E over dataset D, and extends the inital results of P2. They can also state that paper P3 performs a nice literature review.

The latter action provides information that is useful for navigating from

[6] `http://www.youtube.com/user/liquidjournals`

a resource to related resources and therefore to find related information, as shown in figure 2.3.

With the first two actions, the scientific community collectively establish what is worth reading. Feedback in this form is not intrusive but gets beneficiated by actions that are anyways useful for editors or readers. This work of selecting and sharing knowledge is what we do every day. What LJ tries to add is to capture this information by making it easy and convenient for each of us to select and share resources and by then implicitly using the collective (implicit) opinions expressed by people by selection and sharing content. In other words, by giving scientists a tool to collect, organize, and share interesting scientific resources we aim at having a way to assess the interestingness of such resources, and consequently a way to filter interesting knowledge and help manage the information overload. Furthermore, expanding the reach of metrics to other types of content and other activities will allow us to look into other aspects of researcher's productivity. For example, we explore how to reward people sharing good ideas (e.g., by posting them in a blog), selecting and creating good collections of contributions and also giving constructive feedback. Traditional metrics not only are unable to provide such insights but they are still based on citations, which have shown to have flaws [22].

The conceptual model of LJ also provides the information to capture these aspects in the dimensions of the scientific contributions, in the subscription links, in the structural links that make contributions appear in journals, in the usage information (tags, forwarding, sharing). All these rich information gathering aspects are not covered by the traditional model.

From an evaluation perspective, we see the main contribution of this work in providing the basic information for evaluating all sorts of resources based on community opinions implicitly provided. Out of these, many new metrics can be developed, just like many citation-based metrics popped

up once it has been possible to compute citations automatically. A trivial approach consists in counting the number of journals in which a resource appears, or the number of people that shares it, or tag it, etc. A more sophisticated example is provided in [38], where opinions, tags, selection in journals, and other actions that can be expressed via (and recorded by) a liquid journal contribute to the reputation of a resource. This is the algorithm currenlty integrated with the LJ platform.

However, as it is unfeasible to provide a unique (and accepted) magic formula that captures all these aspects, we focus on providing the guidelines that will govern the instantiation of particular derived metrics. Indeed, we believe it is the community to decide what counts within the community. We are developing this concept with the metric uCount[7] that, as the name suggests, captures both the fact that everyone in the community counts and that everyone is involved in the process of defining what counts in the specific community. The idea is that anybody can then decide which metric formula to use to filter out the resources of interest when searching for content on the Web.

2.4 Architecture

Designing and implementing an infrastructure for supporting the LJ model requires solutions and strategies for: i) managing the journals process, ii) journal creation, evolution, consumption and sharing; iii) access to scientific content in the Web, iv) computing the reputation of contribution (for ranking), and the v) projection of these features to a user interface. The LJ architecture relies on specialized components designed for each of the aspects mentioned. In Figure 2.4 we illustrate these components.

Liquid journals[8] provides a view of the scientific content available of the

[7] In joint collaboration with ICST, icst.org

[8] `http://liquidjournal.org`

Web. Since scientific contributions fall outside traditional sources (e.g., digital libraries) where standards can be applied, the infrastructure requires an access layer that provides the necessary abstractions for accessing and searching content on the Web. In order to address this requirement, we rely on the abstraction of a Resource Space Management Systems (RSMS) [5] applied to the scientific domain.

The ResMan[9] system, a prototype implementation of a RSMS, provides a uniform access layer to resources available on the Web. It abstracts applications on top of the heterogeneity of the underlying services on the Web. The approach followed by the system is to rely on adapters, i.e. components that map the specifics of different and non compatible services to a common and uniform protocol [10].

On top of ResMan, the abstraction of a Scientific RSMS[10] (named Karaku system) provides a common and extensible conceptual model specific for scientific resources, and a set of basic services for searching and operating on these resources. A core module is the Updater, which functions as a crawler over scientific sources and extracts resource metadata. This allows us to push resources into the system in the same way users can do it with their iPhone or using the web interface. On these architectural foundation, the liquid journal core component builds the services that support the model introduced in this paper.

LJs allow users to define their own process and to this end, the architecture includes also a lifecycle management component, the Gelee system [6]. The backend is completed by the Research Evaluation tool (Reseval)[11], an extensible tool for computing metrics for contributions and papers (and any other user-defined entity). In this context, the tool takes information about scientific entities from Karaku and applies the algorithms for com-

[9] http://project.liquidpub.org/resman
[10] http://project.liquidpub.org/karaku
[11] http://project.liquidpub.org/reseval

puting metrics. Thus, LJs can be seen as domain-specific mashups that allow users to define the content, process and metrics.

Services are very important in our architecture but to fully exploiting them it is necessary to provide an effective Web interface that facilitates journal definition, search, content consumption and sharing. In our approach, we pay special attention to this issue and we are developing a rich Web application on top of the core components (see Figure 2.5). We also integrated the application with the Facebook social network with the goal of facilitating the sharing, and making it easier for people to use and connect with the system. This is possible due to the Facebook Connect service[12].

2.4.1 Related work

In spite of the progress in dissemination models, the current model of publishing and evaluating scientific contributions remains almost the same. Novel models such as the deconstructed model [49] and the overlay journal [42] introduce interesting ideas yet to be explored and taken beyond structural changes to meet the Web. These models are still constrained to the traditional notion of paper, and so other types of contribution remain hidden. The social part, the study of behaviors that are good for science, such as early feedback, sharing and collaboration remain also unexplored. More importantly, none of the models tackles, and offers mechanism to face, the problem of attention. All these issues affect also the evaluation, which continues to be based on papers (citation-based, e.g., [25][31]) and so leaving out other aspects of research productivity.

The Social Web has made possible new forms of collaboration. Prominent examples are the social bookmarking services that allow users to share

[12] `http://developers.facebook.com/connect.php`

interest within communities. CiteULike[13], Mendeley[14], Zotero[15] and Connotea [16] are examples of social bookmarking services with the focus on sharing and organizing academic references. These tools come with social tagging features that allow people to collaboratively tag content. Thus, these tools provides storing, sharing and tagging of references to publications via shared collections and groups.

Tools for sharing and collaboration stand as a promising direction. These systems provide some foundations and results for further studies in the scientific domain regarding collaboration. However, they are only the "mean" for collaboration without a formal and complete knowledge dissemination model established. Moreover, taking technical aspects apart, one disadvantage of these services is that they rely on active users, that is, users who inject content into the system. Thus, the discovery is limited to what is already there. Our model builds on some social features of these systems but provide a complete model of dissemination designed specially to overcome the dissemination overload in the scientific domain.

Search is a common service on the Web and so search engine technology has been explored and applied to scientific content [34]. Specialized search engines, such as Google Scholar[17] and CiteSeer[18], have been developed for searching papers/books across multiple repositories using crawling techniques and protocols. Using another approach, the academic search engine, BASE[19], indexes the metadata from repositories that implement the OAI-PMH protocol. In addition to what the user can provide as input to the search (e.g. keywords), implicit preferences and collaborative filtering

[13] http://www.citeulike.org
[14] http://www.mendeley.com
[15] http://www.zotero.org
[16] http://www.connotea.org
[17] http://scholar.google.com
[18] http://citeseerx.ist.psu.edu
[19] http://www.base-search.net

has also been used for bringing users content they might like [48]. This has led to general relevance and diversity algorithms proposals trying to balance user preferences and diversity (e.g. [21]). In the academic domain, recommendation of papers have also been explored in many studies (e.g., an evaluation in [41]).

Thus, academic search engines provide only a partial view of the scientific contributions dispersed over several sources on the web. They do not capture the user preferences and lack of proactive behavior. Users need to know what to search and how to search in order to get content, and when they do find an interesting resource, the navigation/exploration is limited. General approaches provide the foundation but their use in the scientific domain need to be modeled for the broader notion of scientific contribution, and other special issues of the scientific domain (e.g., ranking). In our approach we rely on a model that provides semantic relations that can be exploited to explore and discover new similar and related scientific resources.

2.4.2 Conclusion

The Liquid Journal model has been developed in cooperation with Springer and other partners of the Liquidpub project and is being deployed as part of ICST - and, as such, made available to a large community of users. The model enables information filtering through the pillars of a structured (but flexible) model for contributions, a model for journals that exploits the selection capabilities from the community, a consequent metric model, and finally a community discovery approach that identifies scientific communities, maps contributions to communities, and can therefore "suggest" contributions from different communities. Together this can address the information overload problem in science while maintaining the potential that derives from having a lot of information available, that of leverag-

ing breadth in the search. Demos and further details are available from liquidpub.org, the project web site.

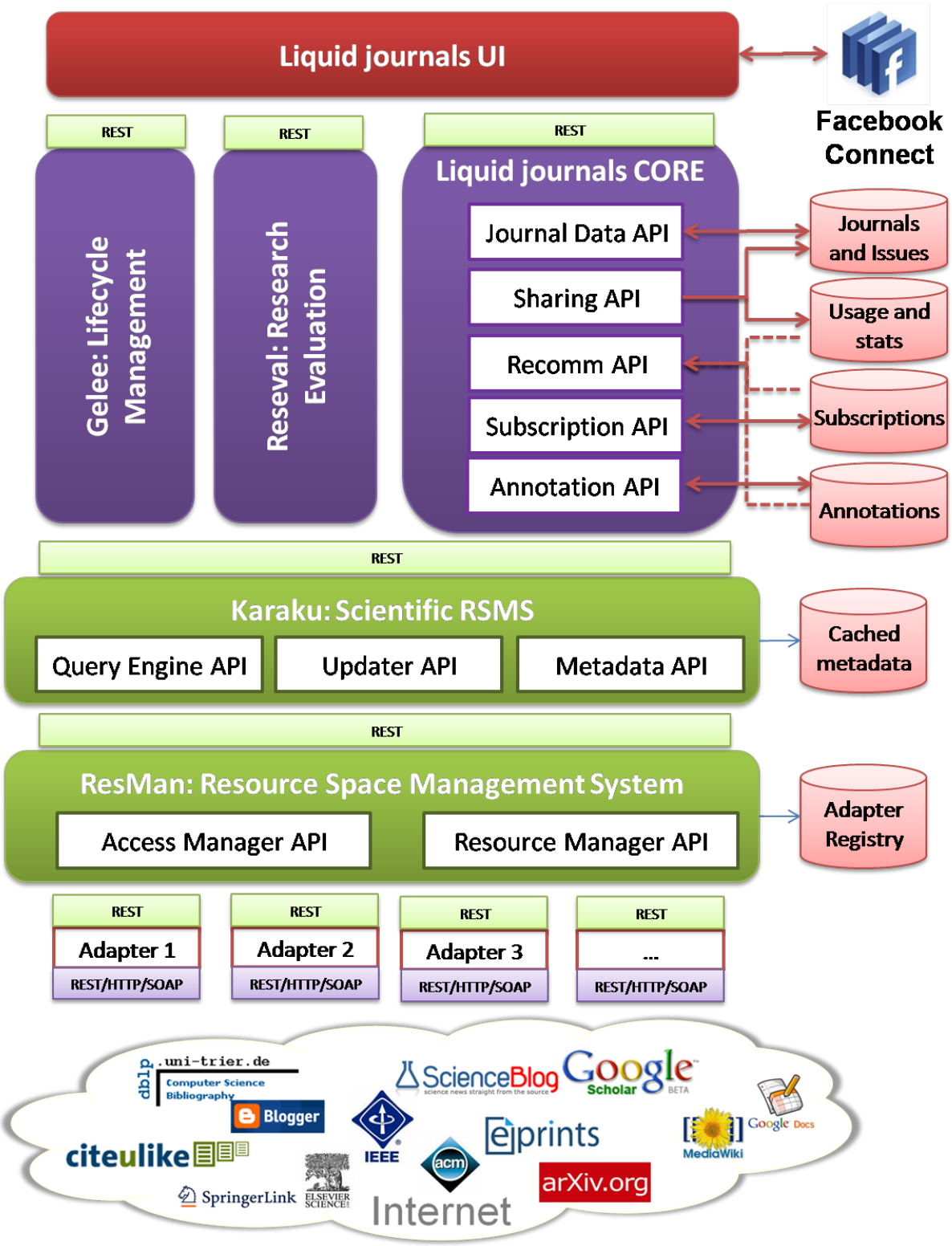


Figure 2.4: LJ backend architecture

Figure 2.5: LJ frontend collage

Chapter 3

Understanding and supporting search for scholarly knowledge

Baez, M. and Mirylenka, D. and Parra, C.

In the last decade, scholarly communication have been greatly transformed by the web, moving research dissemination away from printed papers in journals to digital content that can be easily posted on the Internet. This technical factor along with a larger scientific community makes it really hard to find relevant content for research in the ever growing sea of publications. With the goal of gaining insights on how researchers find relevant knowledge, we have interviewed a small group of researchers and then opened an online survey to a larger group, asking them to explain how they had found references for one of their papers. The results of this study suggest that finding scientific knowledge has a strong social component, with the different researchers' social networks (e.g., coauthors, people met at conferences) accounting for a important percentage of the source of the references. In this paper we report on this study and compare our results with the evidence found in a dataset of 5×10^6 *authors with their publications and references. We take these results and analyze different approaches for incorporating the social component into search and recommendation of*

scientific publications.

3.1 Introduction

The notion of scientific paper as the main means of scientific knowledge dissemination and peer review as the main mechanism to guarantee quality have been, for a long time, the cornerstones of scientific knowledge advance. In the last decades the scientific world has met great changes, the Web being the changing factor pushing us to gradually move away from printed papers in journals to digital content that can be easily posted online. In this context it becomes really hard to find relevant content for research in the ever growing sea of publications. This phenomenon, referred to as "information overload", is a reality and a challenge for the scientific community. It requires an understanding of the problem in this domain and the development of models and tools to overcome its effect.

Motivated by this challenge, we started to study how researchers find scientific knowledge looking for ways to improve the support of this process, taking as a particular use case the problem of finding relevant references. In our study we have found that a third of all the references cited in a scientific paper comes from the authors' interaction with their social networks, including co-authors, project colleagues, and people met at conferences or other events. Researchers stumble upon relevant scientific resources and share them *within* and *among* these different social networks in different contexts. This comes to support the observation that "networking" is highly important in our community. This strong social component, however, has not been fully exploited to overcome the information overload

problem.

Current approaches, such as social bookmarking sites and other social networking systems provide tools and services to share the knowledge within a single context of their systems. However, they do not consider that different incentives and tools are required to effectively capture this knowledge in different contexts. People share, discover and discuss papers, usually informally, at conferences, lectures, in the mailing lists of their research groups and projects. We argue that capturing this knowledge will allow us to understand what people consider important and relevant. The fact, for example, that somebody (and especially somebody we "trust") shares a paper tells us a lot on the value of this paper, more than a citation can do. This fact supports our intuition that, having this kind of information available, we will be able to use it to improve search [24].

In this paper we introduce Knowledge Spaces, an approach to capturing and supporting search for scholarly knowledge. Knowledge spaces (kspaces for short) are a metaphor, a set of models and processes, and a social web platform that help you capture, share and find scientific knowledge in all of its forms. The principle behind kspaces is to allow knowledge dissemination in the scientific community to occur in a way similar to the way we share knowledge with our colleagues in informal settings. The rationale behind this is that when we interact informally with a small team of colleagues dissemination is very effective. We are free to choose the best format for communicating our thoughts and results, we share both established results as well as latest ideas, we interact and carry on a conversation (synchronously or via email), we comment on other people's contributions and papers and observe relations among various contributions.

As regards search, we present our preliminary work towards exploiting this social aspect. Firstly, we discuss how to reuse the valuable social metadata already available on the Web in a scientific metasearch engine.

Then we give some insights on using researcher's social network for search and recommendation of scholarly publications.

The contributions of this work are the following:

- A study on the impact of researchers' social networks in finding references for publications, and on the importance of different kinds of networks,
- A model and a social platform for capturing knowledge and enabling search,
- A scientific metasearch engine leveraging social metrics, and
- Initial ideas on using researchers' social networks, and in particular co-authorship network, for personalized search for scholarly publications.

In what follows we present our study on understanding scholarly knowledge search, describe Knowledge Spaces as its enabling factor, and describe our proposals of scientific metasearch and network-aware search.

3.2 Understanding search

If we are to improve the way we find scientific knowledge, the very first step to do so is to understand how this process naturally works in the mind of those who search. With the goal of reaching this understanding we have conducted a sociological study consisting of two phases:

1. **Qualitative Analysis** of researchers' comments on how they find scientific knowledge they later cite, with the goal of finding an small set of categories in which we could classify this process. For this purpose, we have interviewed 30 researchers from the University of Trento, asking them to explain how they had found references for one

particular publication of their authorship. The answers to the question ranged from "my advisor suggested it" to "I searched for papers in topic X using google scholar" and "I found it while following citation links".

2. **Quantitative Analysis** of an online survey on the same subject, using the categories we have found on the first phase of the study. The online survey[1] would ask researchers to provide their names, which we would later use to search for their publications on a dataset extracted from Microsoft Academic Search[2] of 5×10^6 authors with their publications and references. After selecting one publication, the survey ask the same question as the interview, but this time providing as optional answers the categories we have found on the previous phase.

The results we discuss on the following sub-sections, led us to the conclusion that finding scientific knowledge has a strong social component, which is a clear motivation to incorporate this component in the way we search for scientific knowledge.

3.2.1 Qualitative Analysis Results

Based on the interviews, we have run a qualitative analysis by classifying textual transcripts for each analysed reference. From this analysis, we have classified the sources of references into three main categories:

- **social:** includes all the references that authors came to know thanks to the interaction with their social network including co-authors, project colleagues, people met at conferences or other events;
- **keyword search:** includes all the references found while searching

[1]The survey is still available at http://survey.mateine.org/
[2]http://academic.research.microsoft.com/

for some topics or keywords using tools for that purpose (e.g. google scholar, dblp, specific digital libraries);

- **navigation:** includes all the references found by following citations or other type of references in papers and other resources.

Figure 3.1 shows the average percentage of references in a paper that follows each of the patterns explained before. In general, the same proportion of references comes from a social network of the authors and from specific searches they run on their own, reaching a 38%. References they got from navigating through knowledge represent the remaining 24%. When divided by seniority, results hold the same trend, with the social scoring higher than search both for professors and postdocs, and lower only for Ph.D. students. The later can be seen as a very intuitive result taking into account that the academic social network of a Ph.D. student is typically smaller than that of a senior researcher.

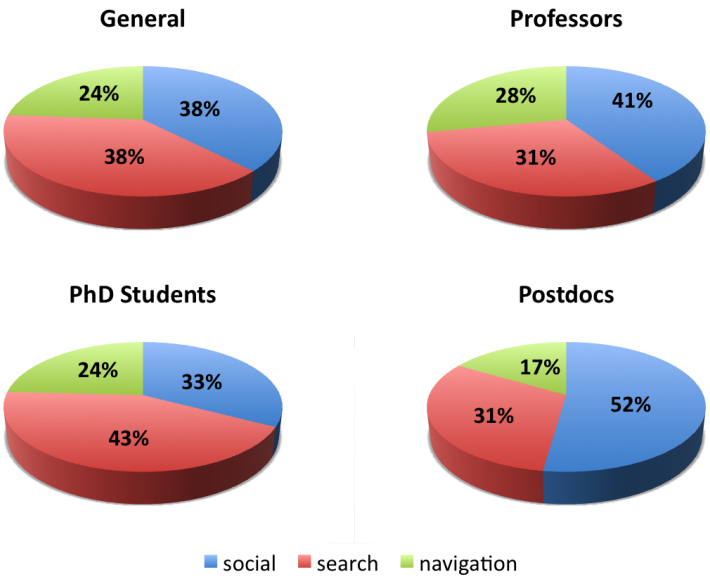


Figure 3.1: Sources of references in a research paper, classified by seniority of the researcher

Furthermore, and given the high percentage of social related references, we have also classified the networks most commonly mentioned as sources of references. Figure 3.2 shows how many of the social references correspond to each of the following networks:

- **community/field** includes references that come from people or projects that can be considered as part of the same field or community around a certain topic, but whom the author has not necessarily met;
- **colleagues**, including peers whom the author has appointed as such. This is a very general term used by most of the interviewees and that might have a high intersection with other networks;
- **venues** includes references the author came to know through conferences or journals.
- **collaboration**, including people, groups or projects with whom the author has directly collaborated;
- **senior colleagues** includes mainly advisers or experts in a specific field;
- **coauthors**, including people who coauthored at least one article with the interviewee;
- **research group** includes people working in the author's department or research group;
- **acquaintances**, including people the author has personally met, which is also a general term used by interviewees that could also be included in other networks;
- **friends** includes people specifically appointed as such by the interviewee

- **educational** includes references the author got from courses, classmates or education related networks;

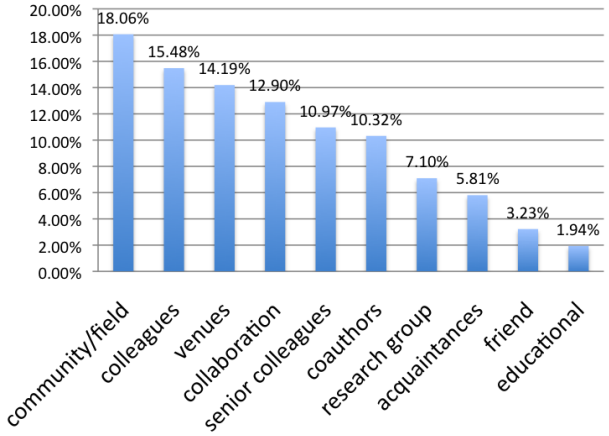


Figure 3.2: Social networks of origin for references

In the same way, most citations found by navigation, were found while reading a paper or book, and then going deeper in the citation graph. Other subcategories of the navigation pattern include the follow up of one particular author, journal, conference or project. As for references found by keyword search, google and google scholar are the most used engines, while also dblp was mentioned (probably, due to the high number of computer scientist in the group of interviewees).

At the end of the study, we decided also to ask which of the analyzed references in the interview they liked the most. Even though this question was not in the original interview script, the trend we have found and that would later be confirmed by the online survey is that most liked references come mainly from authors' social networks, accounting up to a 41%.

Data: The analysis of this phase of the study is based on 351 references (without counting 43 self citations) spread across 30 interviews (18 Ph.D. students, 8 postdocs, 4 professors). Of these, 214 correspond to Ph.D.

students, 64 are from professors and 71 from postdocs. We have removed self-citations from the analysis as the authors already knew them and had no need to find them. The interviews were conducted during May of 2011 resulting in 789 different notes that were later manually categorized to get the before mentioned results.

3.2.2 Quantitative Analysis Results

The second phase of our preliminary study consisted on conducting an online automated version of our interview, using the categories we have found on the first analysis.

Figure 3.3 shows the average percentage of references in a paper in each category. The numbers are similar to those of the first phase, with the exception of a significant drop in the percentage corresponding to navigation while the number of references for which authors selected the option **Do not remember** increased dramatically (especially for professors).

The reason behind the decrease in the navigation references percentage could be that the explanation in the online survey was not clear enough. Other reason might be that we are missing an important category, which we expect to discover as more people participate of the survey and provide feedback on possible missing categories.

For each of category, the online survey also included the option to further detail the answer by indicating

1. the social network from where the reference came from (for the social category),
2. the search engine or repository (for keyword search), and
3. while navigating what (for navigation)

Figure 3.4 shows the percentage of social references by social network. Although the community is again the mos important network, there is a

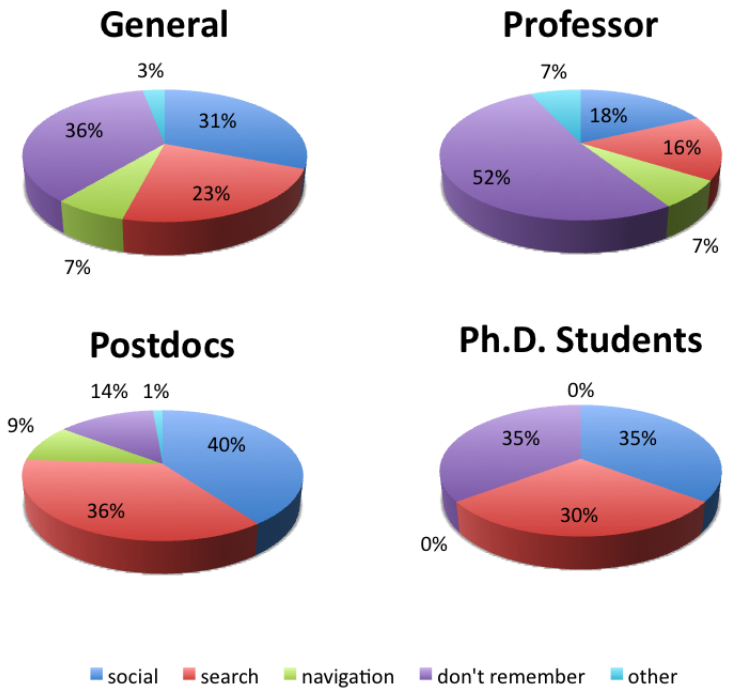


Figure 3.3: Results of our online survey, classified by seniority of the researcher

significant increase in the percentage corresponding to coauthorship, which is not in line with the first phase analysis, implying that more research needs to be done in order to improve our understanding about the relevance of each of the many different social networks of a researcher. This however, we have gained interesting insights about which networks are the ones we have to investigate further.

More details and results of this analysis are available online and will be constantly updated in the site of the survey[3].

Data: The online survey, currently ongoing, has gotten to this date, the reply of 28 different researchers, that responded about the source for a 226 references distributed over 23 different publications. Aggregating these responses confirm what we have already found in the interviews: the social

[3]http://survey.mateine.org/results

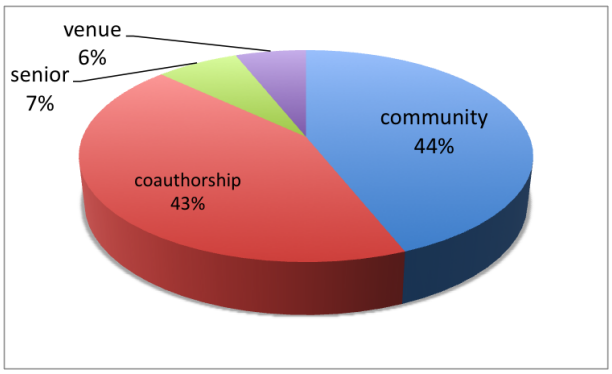


Figure 3.4: Social references by social network

component is stronger than any other.

3.3 Defining Network-aware recommendations

The goal of this work is to incorporate the social component of knowledge discovery into recommendation of scientific publications. More specifically, we aim to build a recommender system that suggests publications that researchers are likely to find through their social networks and that are related to their work.

Our model represents the graph of researchers and scientific publications connected with the relations of authorship and citation. We formalize the problem definition in the rest of the section.

3.3.1 Formal problem definition

Let $\mathcal{R}$ be the set of all researchers in the system, and $\mathcal{P}$ - the set of all publications. Relation $\mathsf{Authored}$ is defined by the set of pairs $(researcher, publication) \in \mathcal{R} \times \mathcal{P}$ such that $researcher$ authored $publication$. Similarly, Cited includes all pairs of publications $(citing, cited) \in \mathcal{P}^2$ such that $citing$ references $cited$. Publication p is cited by researcher r if it is cited by any paper of r:

$$\mathsf{Cited}(r, p) \iff \exists p' \in \mathcal{P}\,(\mathsf{Authored}(r, p') \wedge \mathsf{Cited}(p', p))\,.$$

For each researcher r we define a set of known and a set of unknown publications:

$$\mathsf{Known}(r) = \{p \in \mathcal{P} \mid \mathsf{Authored}(r,p) \vee \mathsf{Cited}(r,p)\},$$

$$\mathsf{Unknown}(r) = \mathcal{P} \setminus \mathsf{Known}(r).$$

With each researcher r we also associate a network, which is set of researchers similar to r according to some similarity function:

$$\mathsf{Network}(r) = \{r' \in \mathcal{R} \mid \mathrm{sim}(r,r') > \delta\}.$$

Examples of such a network may include coauthors of r, or researchers publishing in the same conferences, or researchers citing the same papers.

The popularity of a publication p within a set of researchers s is defined as $\mathsf{popularity}(p,s)$ and its definition depends on the particular recommendation strategy. We introduce some strategies in the next subsection.

Given the definitions above, we formulate the problem of social recommendation of scientific publications: For a given researcher, find k publications unknown to him/her that have the highest popularity in his/her network:

1. $\mathsf{Rec}(r,k) \subseteq \mathsf{Unknown}(r)$,
2. $|\mathsf{Rec}(r,k)| = k$,
3. $\forall p \in \mathsf{Rec}(r,k)\ \forall p' \in (\mathsf{Unknown}(r) \setminus \mathsf{Rec}(r,k))$
   $(\mathsf{popularity}(p, \mathsf{Network}(r)) \geq \mathsf{popularity}(p', \mathsf{Network}(r)))$.

In what follows we explore different definitions of network and popularity to later evaluate and analyze their performance.

3.3.2 Defining the notion of network

Recommendations for scholarly knowledge depend on the context and the goal the user is trying to achieve. They are also strongly related to the

type of network and the algorithms used to compute them. In this section we focus on defining different network configurations around researchers.

Coauthorship network

We first introduce the co-authorship network based on our definition of network :

$$\mathsf{Coauthors}(r) = \{r' \in \mathcal{R} \mid \mathrm{sim}(r, r') > \delta\}$$

expressing in the similarity function the number of papers two researchers have written together normalized by the number of publications, and then applying $\delta > 0$ to create the network of all the coauthors a given researcher:

$$\mathsf{sim}(r, r') = \frac{\|\mathsf{Publications}(r) \cap \mathsf{Publications}(r')\|}{\|\mathsf{Publications}(r)\|}$$

where

$$\mathsf{Publications}(r) = \bigcup \{p \in \mathcal{P} \mid \mathsf{Auhtored}(r, p)\}\,.$$

Venue network

Our definition of venue network tries to capture the likelihood of two researchers meeting at a venue, resulting in a future citation. We define the relation $\mathsf{VenueOf}$ as a set of pairs $(\textit{publication}, \textit{venue}) \in \mathcal{P} \times \mathcal{V}$:

$$\mathsf{Copublished}(r) = \{r' \in \mathcal{R} \mid \mathrm{sim}(r, r') > \delta\}$$

where the similarity function expresses the normalized number of venues two researchers have published together:

$$\mathsf{sim}(r, r') = \frac{\|\mathsf{Venues}(r) \cap \mathsf{Venues}(r')\|}{\|\mathsf{Venues}(r)\|}$$

given that

$$\mathsf{Venues}(r) = \bigcup \{\mathsf{VenueOf}(p \in \mathcal{P}) \mid \mathsf{Auhtored}(r, p)\}\,.$$

Topic network

Topic-based network capture the notion of researchers working in the same field. We assume each publication p belongs to a set of topics $\mathsf{Topics}(p)$ thus:

$$\mathsf{Co-topic}(r) = \{r' \in \mathcal{R} \mid \mathrm{sim}(r, r') > \delta\}$$

where the similarity function expresses the normalized number of topics on which two researchers have both published:

$$\mathsf{sim}(r, r') = \frac{\|\mathsf{Afinity}(r) \cap \mathsf{Afinity}(r')\|}{\|\mathsf{Afinity}(r)\|}$$

given that

$$\mathsf{Afinity}(r) = \bigcup \{\mathsf{Topics}(p \in \mathcal{P}) \mid \mathsf{Auhtored}(r, p)\}\,.$$

3.3.3 Defining popularity functions

On the above we define different popularity functions that explore different views on the importance of a publication in the researcher's network:

- *network popularity*: The popularity of a publication p within the network of a researcher r is defined as the number of researchers in p who either authored or cited p:

$$\mathsf{pop_n}(p, r) = \frac{\|\{r' \in \mathsf{Network}(r) \mid p \in \mathsf{Known}(r')\}\|}{\|\mathsf{Network}(r)\|}.$$

- *work-weighted network popularity*: Expresses the popularity of a publication p in the network of a researcher r, weighted by her similarity with all researchers in the network:

$$\mathsf{pop_{wk}}(p, r) = \frac{\sum \{\mathsf{sim}(r, r' \in \mathsf{Network}(r)) \mid p \in \mathsf{Known}(r')\}}{\|\mathsf{Network}(r)\|}.$$

- *time-weighted network popularity*: Expresses the popularity of a publication p in the network of a researcher r, weighted by temporal similarity with other researchers in the network, considering the range $[y_{min}, y_{max}]$:

$$\mathsf{pop}_{\mathsf{wt}}(p, r) = \frac{\sum \{\mathsf{sim}_{\mathsf{t}}(r, r' \in \mathsf{Network}(r)) \mid p \in \mathsf{Known}(r')\}}{\|\mathsf{Network}(r)\|}$$

- *overall popularity*: The overall popularity of a publication p is defined as the number of all researchers who either authored or cited p:

$$\mathsf{popularity}_{\mathsf{o}}(p) = \frac{\|\{r \in \mathcal{R} \mid p \in \mathsf{Known}(r)\}\|}{\|\mathcal{R}\|}.$$

Recommending for a topic

The problem definition formulated above can be extended to the case where the recommendations are restricted to specific topics of interest. We assume each publication p belongs to a set of topics $\mathsf{Topics}(p)$. Let $\mathsf{FilteredBy}(ts)$ be a set of publications belonging to at least one topic from ts:

$$\mathsf{FilteredBy}(ts) = \{p \in \mathcal{P} \mid \mathsf{Topics}(p) \cap ts \neq \varnothing\}.$$

For a given researcher r and a set of topics ts, we need to find a set of publications $\mathsf{Rec}(r, ts, k)$ such that

1. $\mathsf{Rec}(r, ts, k) \subseteq \mathsf{FilteredBy}(ts) \cap \mathsf{Unknown}(r)$,
2. $|\mathsf{Rec}(r, ts, k)| = k$,
3. $\forall p \in \mathsf{Rec}(r, ts, k)$
   $\forall p' \in (\mathsf{FilteredBy}(ts) \cap \mathsf{Unknown}(r)) \setminus \mathsf{Rec}(r, ts, k)$
   $(\mathsf{popularity}(p, \mathsf{Network}(r)) \geq \mathsf{popularity}(p', \mathsf{Network}(r)))$.

This extended problem definition will make possible the recommendations on a topic and the implementation of a network-aware search for scientific publications. This should be accomplished by mapping the search query specified by the user to the set of topics, and recommending the publications for this set of topics based on the user's network. In this work, however, we don't address the problem of topic-based recommendation.

3.4 Evaluation

In this section we present the evaluation of the social recommendations we introduced in the previous section.

3.4.1 Experiment definition

We obtained a crawled copy of the academic search database[4] containing data about publications, their authors and citation relations between them.

Our goal was to evaluate the ability of our recommender system to produce relevant recommendations for researchers depending on different popularity functions introduced in Section 3.3.3.

For the purpose of this experiment we assumed citation to be the indication of relevance. In other words, we considered publication p to be relevant for a researcher r at some moment in the year y if r cited p after the year y. We then evaluated the precision and recall of our recommendations. This evaluation was done by measuring how well our algorithms predicted researchers' citations after the year y based on citations of their network before that year. For a random sample of 1000 researchers we ran the experiment for different combinations of year y (ranging from 1999 to 2009), number of produced recommendations (1, 2, 4, 8 and so forth, following an exponential growth up to 512), and popularity function (Section

[4]http://academic.research.microsoft.com/

3.3.3), averaging the precision and recall metrics over the researchers in the sample.

3.4.2 Dataset Description

Our dataset contained 7.465.398 unique publications written by 5.726.226 different authors. As the design of our experiment required the year of publication to be known, we selected 3.937.907 papers for which this information was available. In order to improve the dataset, we approximated the publication year for 1.907.589 more publications by taking the maximal year of their references. Hence the total number of publications participating in experiment was 5.845.496.

3.4.3 Running and analyzing the experiments

Before running the experiment as described in the previous subsection, we analyzed the inherent quality of the coauthorship network as source of recommendations. According to our dataset, 20% of future citations for a researcher overlaps with the past citations from her coauthorship network. This percentage represents the maximum recall that any of our popularity functions could achieve.

In Figure 3.5 and 3.6 we present the precision and recall for each popularity function and year-cut.

In all methods the tendency points to a decrease in the precision by the year. This owes in part to the distribution of the dataset, but more importantly to the fact that the set of future citation declines with the year, decreasing the maximal number of relevant publications. In the case of the recall, the results are not necessarily related to the year, but to the number of recommendations.

Analyzing the performance in terms of number of recommendations,

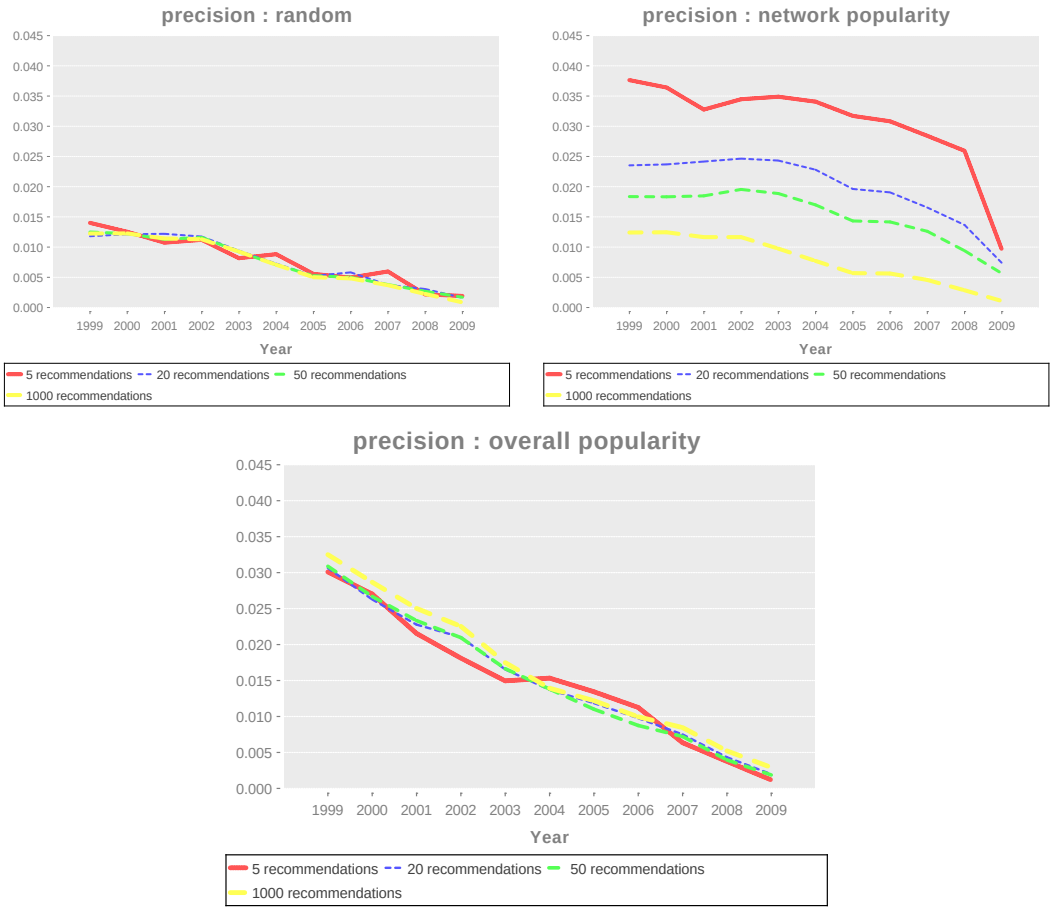


Figure 3.5: Precision of the different popularity functions by year

we can see that the precision of network-aware popularity gets better as the number of recommendation decreases. It means that the papers most cited by the researcher's network are much more likely to be relevant to the researcher. This effect requires further investigation in order to be fully explained. However, our preliminary hypothesis is that it may be due to the fact that the most cited papers in the researchers network belong to the topics relevant to the whole community of this network (which explains many citations) and thus likely to be relevant to the researcher, while the less-cited papers have topics relevant only to a part of the community (therefore, having smaller number of citations) to which the researcher is

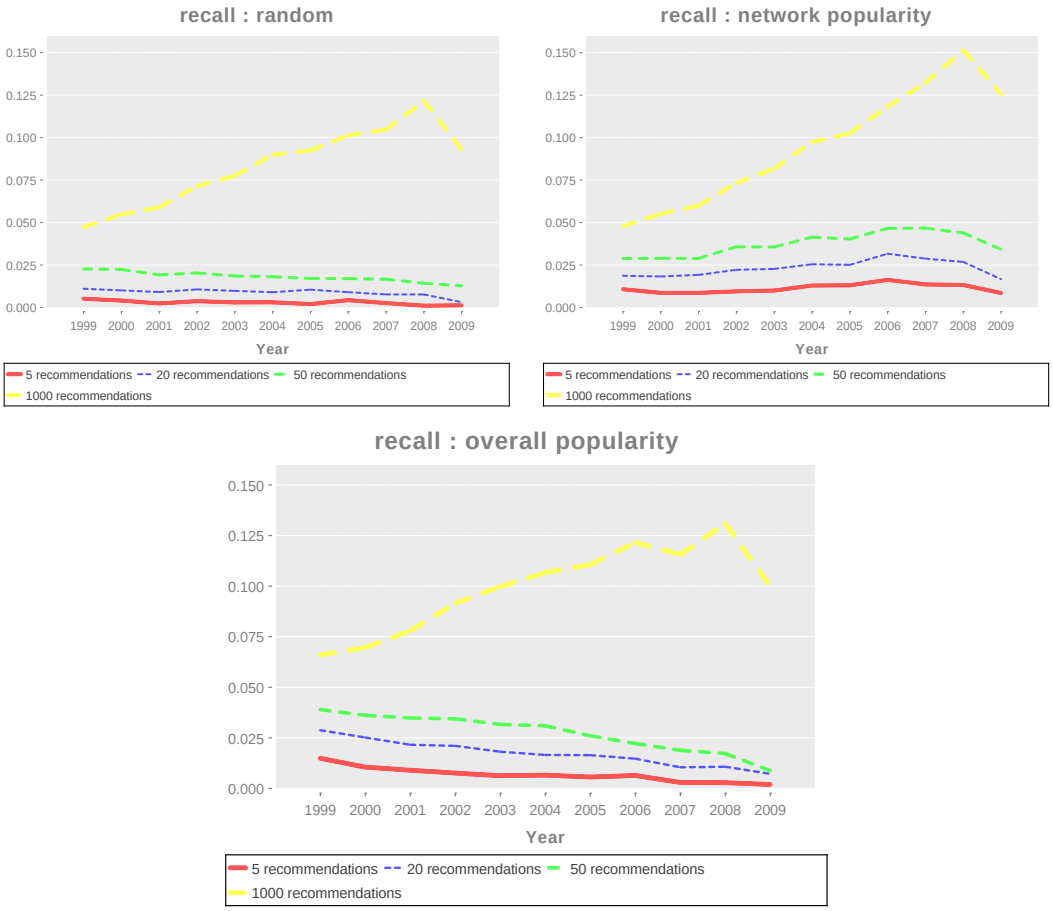


Figure 3.6: Recall of the different popularity functions by year

less likely to belong. This also explains why network-aware popularity outperforms random and overall popularity, especially for the small numbers of recommendations.

The random popularity shows very low precision but we can see that it is not sensitive to the number of recommendations. This can be explained by the fact that percentage of relevant papers in the random sample of papers does not depend on the sample size. The overall popularity function seems insensitive to the number of recommendations in terms of precision too, while explaining this phenomenon needs further analysis. We can also see that there is always a quality measure in the selection of references, and

this becomes evident in how the overall popularity outperforms the random popularity.

The recall of the three described popularity metrics shows the strong tendency to grow as the number of recommendation increases, with the network-aware recommendation generally outperforming the other two methods. The difference between the performances of the three methods in terms of recall is the largest on the small number of recommendations and almost vanishes as the number of recommended papers reaches 1000. This is due to the fact that the different rankings of the publications (known to researcher's network) do not change set of recommendations as the number of recommended papers tends to the total number of papers in the network.

Finally, the fact that the popularity in the network performs better than the overall is an indicator of the importance of considering the network.

3.4.4 Discussion

Our results allow us to infer that the social awareness approach can provide some improvement in the recommendation of scholarly publications, but further exploration is still needed to understand how different networks would affect the results and, in particular, which of these networks (or combination of) have the best recommending power. It is interesting to see that the average recall of our co-authorship network-aware algorithm (being in the range of 1 and 15 percent) is similar to the percentage of papers coming from this network according to our study (almost 11% of the overall 40% of social-originated references).

Furthermore, a major improvement to be made is to include topic analysis within our recommendation logic. Our intuition is that such a logic can generate a better and more relevant final ranking of resources. More experiments also need to be performed to find relevant citation patterns in the networks we have available in our dataset.

Finally, the ranking scores we have used in this work were chosen for their relative straightforward implementation, while more complex ranking scores remain untested. Ranking recommendations following some notions of weighted co-authorship (based either on the number of coauthored resources or the recency of the last collaboration) might provide better results. Unfortunately due to both a lack of time and the characteristics of the available dataset, such rankings remain untested.

A beta prototype of our recommendation system is available for testing and playing at `http://discover.mateine.org/`.

3.5 Related Work

Classical research on user context and search tasks focused on understanding the patterns used in web search [14], resulting in well known taxonomies. Recent research focused on the final user goal on underlying the search, defining classification closer to the user needs [46]. In our study, however, we go domain-specific trying to understand how researchers find scientific knowledge, focusing on the impact of the social aspect.

As for capturing knowledge sharing, an increasing number of social bookmarking and annotation services have become available in the scientific communities. Zotero, CiteULike, Connotea, Mendeley are examples of scientific social bookmarking services that focus on sharing and organizing academic references. These tools and services deal with sharing and collecting materials targeting groups and individuals. However, they are of general purpose, and thus, their effective usage is limited to a reduced number of scenarios. In Knowledge spaces, we take a different approach going vertical to every scenario in order to lower down the barriers to share. An interesting work in this line is Mail2Tag [35], a system that explores the use of mail as a tool for sharing and organizing news in environments

where the email is the main communication channel.

Social bookmarking sites provides an interesting direction, trying to incorporate the social aspect to collaboratively find relevant information. Studies has been carried out, analyzing the use of this social data to improve search [24]. These studies suggest that, having good size and distribution, tags and bookmarks can have an important impact when used in combination with search.

Another direction of research focuses on providing recommendation of scholarly publications. CiteULike employs collaborative filtering in order to suggest publications that might be interesting to users [13] based their reference libraries. [51] reports on a citation recommendation algorithm that relies on publication texts and citation graph to compute the similarity between the references provided by the user and those recommended by the system. A topic-based recommendation system that also uses citation graph is described in [53]. The system described in [1] exploits the user similarity based on their search queries in order to produce recommendations of scientific papers. These approaches are relevant to our problem of network-aware search for publications. Our approach, however, uses the domain-specific social networks of researchers, such as co-authorship or conference-based network, for suggesting relevant publications. We should also note that our approach does not rely on similarity between the publications.

Other studies consider the user social network to provide personalized search results (e.g., [61] [16]). The most relevant to our work is [61], in which the authors propose a network-aware search for social bookmarking sites. This work introduces interesting techniques that can serve as baseline to this project, however, the modeling, analysis and optimization are specific to this domain, and require particular attention.

3.6 Concluding remarks

In this work we have proposed the personalized approach to recommending scientific publications based on researchers' social network. We formalized the problem, considering different definitions of networks and popularity metrics and formulated its topic-based version. Given the dataset of Microsoft Academic Search, we designed and conducted the validation experiment by evaluating precision and recall of three different recommendation strategies within our proposed approach with respect to researchers future citations. We analyzed the results, drew some preliminary conclusions regarding the applicability and the potential of network-based recommendation of scientific publications, and identified directions of future work. Finally, we implemented the recommender system relying on users' co-authorship network and deployed it within the prototype web application.

Chapter 4

Sharing Scientific Knowledge with Knowledge Spaces

Baez, M. and Casati, F. and Marchese, M.

This paper presents a set of models and an extensible social web platform (namely, Knowledge spaces) that supports novel and agile social scientific dissemination processes. Knowledge spaces is based on a model for scientific resources that allows the representation of scientific knowledge and meta-knowledge, of effective viral algorithms for helping scientists find the knowledge they need, and of interaction metaphors that facilitate its usage. The concept and a preliminary implementation of Knowledge spaces, in their various forms and designs, are being exploited in several different pilots in cooperation with IEEE, the EU Commission, Springer, the archeology museum in Cambridge and major international conferences to support the collection and sharing of knowledge in scientific communities.

4.1 Introduction

Knowledge spaces (kspaces for short) are a metaphor, a set of models and processes, and a social web platform that help you capture, share and find scientific knowledge, in all of its forms.

The principle behind kspaces is to allow knowledge dissemination in the scientific community to occur in a way similar to the way we share knowledge with our colleagues in informal settings. The rationale behind this is that when we interact informally with a small team of colleagues dissemination is very effective. We are free to choose the best format for communicating our thoughts and results, we share both established results as well as latest ideas, we interact and carry on a conversation (synchronously or via email), we comment on other people's contributions and papers and observe relations among various contributions. Even when we remain in the domain of papers, we often find that we come to know interesting papers not by doing a web search or scan the proceedings, but because we "stumble upon" them, that is, we have colleagues pointing them to us via email or mentioning them in a conversation (along with their comments), and knowledge spreads virally.

Kspaces aim at providing the models, processes, metrics and tools to support this informal and social way of disseminating knowledge among the scientific community at large and via the Web, complementing the well-established method of papers published in conferences and journals after peer review. The goal is to use a web-based system to enable the capturing of these evolutionary bits of knowledge and data, however they may be expressed, as well as the capturing of ideas and opinions about knowledge, and leverage this information and meta-information to spread knowledge socially. Capturing opinions on knowledge is particularly important. The fact for example that somebody (and especially somebody we trust) shares

a paper tells us a lot on the value of this paper, much more than a citation can do. As readers, we relate them, in our mind, with prior knowledge. When listening to a talk we think that other work is relevant to the one being presented and often we jot it down in our own personal notes. In a world where information comes out from the web like from a hose, this knowledge about knowledge becomes essential to dissemination. Tagging, annotating and connecting the dots (linking resources in a way much more useful to science than citations) become almost as important as the dots themselves.

Kspaces support this not only by using web technologies as the basis for its implementation but by using web 1.0 and 2.0 concepts in the way scientific resources and their relationships are modeled and in the way knowledge sharing is supported. In essence, kspaces is characterized by a conceptual model and a repository for scientific resources (or for pointers to them if stored elsewhere). Resources are linked in arbitrary ways and relationships are typed and can be annotated. This is analogous to the Web, although it is oriented to linking scientific resources and to supporting (and then leveraging) relationship types and annotations. Indeed building this evolving web of annotated resources and leveraging it to find knowledge is a key goal of kspaces. The intuition is that having such web of connected knowledge can be as instrumental or even more instrumental (because it contains more metadata) to finding knowledge than the Web is to finding web pages. Today this web of resources is simply not there and this is part of what makes finding interesting scientific knowledge hard.

On top of this space of resources, kspaces define specific processes, permissions, and interaction modes people use to share knowledge. Kspaces manifest themselves in various forms, called designs, tailored at capturing different forms of scientific knowledge shared in different ways, from maintaining a library of related work, talks, datasets, etc, in an area including

our own, evolving work - to forming knowledge communities, writing and publishing (liquid) books, supporting the collection of the knowledge that emerges in the brain of attendees during a talk, and many others. It is through spaces with specific design that knowledge and meta-knowledge is collected and disseminated. The dissemination and search of knowledge over kspaces is then based on the social interest, on the goals of a search (e.g., related work vs introductory material), and on the meta-knowledge (e.g., tags and annotations). Kspaces, although being richer and more flexible than many existing systems, is not the first and only platform that exploits some form of social meta-knowledge to support search. Mandeley, citeUlike, and Connotea, just to name a few, all have some elements of this. We believe that the key to a successful platform here lies in how such meta-knowledge can be collected and how it is used, and here lies a key contribution of kspaces. We discuss how the state of the art influenced (and differs from) kspaces later in the paper. Kspaces are not aimed at supporting collaborative editing: in other words we do not provide tools in the style of google docs or latex+SVN to allow people to write and extend an idea in a collaborative fashion. The goal is to support the dissemination of such ideas and knowledge. A system that aims at social dissemination would invariably have to face the following social and technological challenges:

- Usage: Researchers are often way too busy, lack incentives, or might even be uncomfortable to go on a web site to tag or comment on knowledge or recommend interesting content to others outside the close circle of colleagues or friend

- Bootstrapping: As in any crowdsourcing system, there is the issue of how to get people to begin to use the system, as only through participation does the network becomes interesting.

- Overload: If the issue of bootstrapping and getting people to share is solved, then the information overload problem appears. As scientists, we are already flooded with large number of papers that sometimes makes it hard to find interesting contributions. If there is even more knowledge shared in more forms, the risk is to just make the problem worse

- Identifying the right models and algorithms, architecture and interaction designs: kspaces depends on a model for scientific resources that allows the representation of scientific knowledge and meta-knowledge, of effective viral algorithms for helping scientists find the knowledge they need, and of UI and interaction metaphors that facilitate its usage.

The contribution of this paper is to present a set of models and an extensible web-based system that aims at overcoming these challenges. We will get back to these challenges throughout the paper and revisit them in the conclusions to see how kspaces can address them. The concept and a preliminary implementation of kspaces, in their various forms and designs, are being exploited in several different pilots in cooperation with IEEE, the EU Commission (who used it at their flagship event for future and emerging technologies, fet11.eu), Springer, the archeology museum in Cambridge and major international conferences to support the collection and sharing of knowledge in conferences, in technical communities, among scholars visiting museums, and in the generation of teaching material among groups of lectures. As discussed later, part of kspaces is now in production and available for general use.

4.2 Knowledge Spaces

Scientific resources - beyond the requirements in terms of structure - serve a specific purpose: they communicate and transfer knowledge in the scientific community. This dissemination process has different components and particular requirements in order to be effective. Kspaces represents the abstraction in which scientific resources are organized, shared, consumed, with the goal of making the dissemination process more effective. It puts a context in all the social interactions and provides the tools that define its dynamics. In the following, we first present a scenario to help us introduce the concepts, taken from one of our pilot applications, Instant Communities. Then we describe the model we envision for representing scientific resources and then discuss what kspaces and designs are and how they can be created on top of a scientific resource space.

4.2.1 Instant Communities scenario

Today, when there is a panel or paper session at a conference, interested people join in. After the panel, the insights from panelists and speakers remain, but most of the thoughts and suggestions of the attendees, which is a great wealth of potential insights for everybody interested in the topic, remains in the mind of the attendees. The temporary community that was created by the event and by the people sitting in the same room is quickly dissolved at the end of the panel. Even very simple knowledge sharing tasks, such as going back and finding the panelists slides to share them with colleagues, are rarely done unless we are really committed to that (which raises the effort barrier and reduces the chances well ever do that). There is often no easy way to follow-up with panelists.

The Instant Communities kspace provides an IT infrastructure that helps create a community of interest in real-time during the panel or ses-

sion. Initially, material is created and posted before the panel, by the panelists. This is an immediate body of knowledge that can be shared among panelists and participants. Then, during the panel, attendees, while listening, if they have a tablet or laptop avail, they can add papers, comments, questions, slides, links, interesting datasets, and whatever they feel useful. The key here is to make it extremely easy for people to add content as attendees are there to listen and interact as primary goal though the instant community can complement this interaction. Another goal is to make it possible to collect questions from the audience and have people vote on questions (this is important to also allow shy people to ask questions), or to have attendees add comments to specific topics or slides being presented. This has specific implications on the interaction design for the during-the-panel phase. After the panel the goal of instant communities is to facilitate collection and sharing of material, to keep the attendees in touch, and extend the community with other people interested. People can also create their own view on this body of knowledge, with a few clicks and drag and drop. One can do so by explicit selection or by filtering by poster, topic, and the like.

They can then share this view, or the entire space, with their team at home, with colleagues, with the entire instant community, etc. Incidentally, all this adding, selecting, and sharing knowledge provides an implicit way to connect people, connect knowledge, and identify interesting knowledge (by looking at what people share). It is a way therefore to provide information that can be used for facilitating search and for assigning reputation to scientific resources. The UI here is focused on ease of searching and browsing and of making it easy for people to share subset of the content with their colleagues. The distinguished role and material of the panelists loses its predominant role as the community takes over.

The detailed list of features, user stories, screenshots and implementa-

tion details of instant communities are available at `http://open.instantcommunities.net`. The application has been used in various conferences and seminar series and will be deployed this summer for intra-company usage. It is one of the way in which kspaces tackle the challenges of bootstrapping and of usage: by providing knowledge capturing and sharing applications for specific purposes and communities. We will see other applications later in this paper.

4.2.2 Scientific Resource Spaces

We see scientific contributions as a structured, evolving, and multi-facet objects. Specifically, we see the space of scientific content we want to collect, organize, share, evaluate, and search as consisting of scientific resources, organized as set of nodes in a graph, that can be connected and annotated by authors or readers. The reasons for connections, and hence for modeling resources as a graph, is to capture several kinds of dependencies or relationships among them. All annotations and relationships - including the ones among resources and contributors and therefore including authorship - can be typed and can be (and typically are) subjective, representing the opinion of a person or of an institution. For example, relations can denote citations and authorships (as in todays papers), but can also indicate versioning, alternative representation of the same content (e.g., a paper and a presentation), usage of datasets (e.g., to state that paper P describe experiment E executed over dataset D) and so on. Types and relationships are arbitrarily extensible by the various kspace designs and their semantics is given within the kspace. For example, a relation table of contents for between two resources is interpreted and visualized in a specific way by the liquid book kspace, while it may be interpreted as a generic relation without any specific UI representation by other designs.

The reason for allowing anybody to define relationships is because in

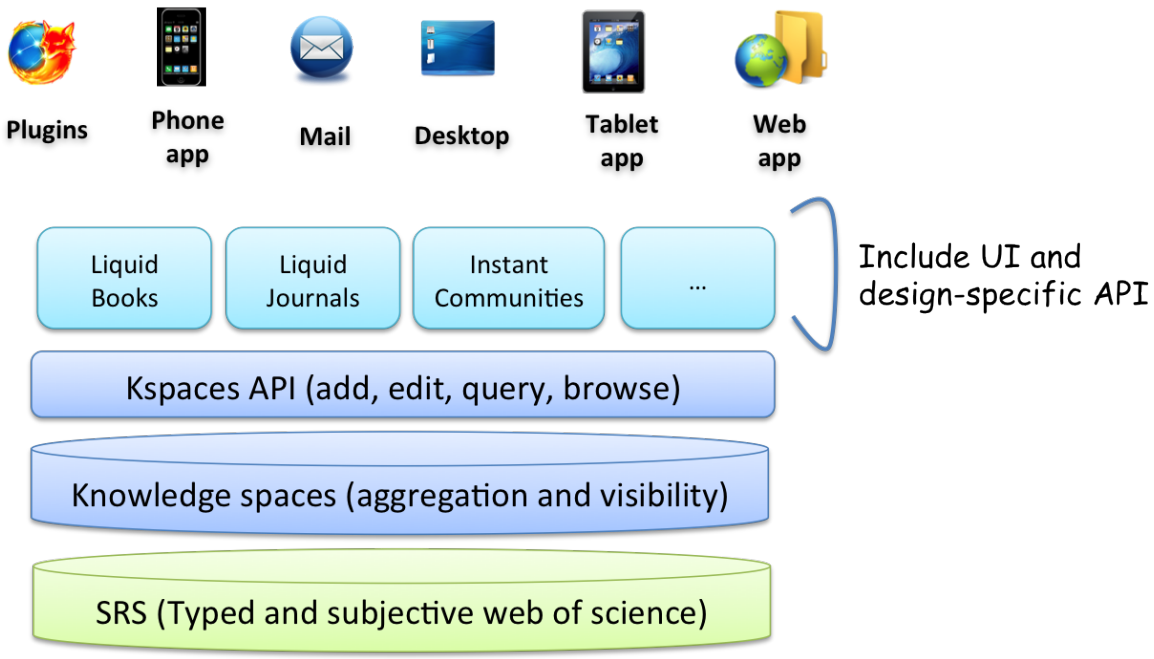


Figure 4.1: Kspaces and KS designs are ways to create, share and consume resources.

this way we can leverage the power of the community to build scientific dissemination knowledge, that is, knowledge that can help annotate and relate resources above and beyond what authors would do. In other words, people generate knowledge that helps in organizing and finding scientific resources.

We do not discuss or formalize the SRS model further as it was discussed in our earlier work [4] to which we refer the reader for details.

4.2.3 Kspaces Conceptual Model

A Knowledge Space is defined as $KS = \{R, Q, M, Tr, C, S\}$, i.e., a collection of SRS content (Figure 4.1), with the following characteristics:

- The content is defined intensionally (in terms of the properties the content should have) or extensionally (content is explicitly added). A space can be only intensional, only extensional, or a mix. In case the

content is defined intensionally, KS defines in essence a query over the SRS, denoted as Q. Explicitly added resources are denoted by R. The intensional language is discussed later in the paper.

- A KS has members $M = \{O, E, V\}$ that can be owners O, editors E, and viewers V. Viewers can only access the resources. Editors can add or remove content. Owners are editors and can add new viewers or editors or owners.

- $Tr = \{transparent|opaque\}$ denotes the transparency flag. A frequent desire when creating a space is to keep the posted resources and/or, most importantly, the comments on them, private to the users of a space. An opaque space is a space where the comments, tags, annotations on resources, and the existence of the space itself are only visible to the members of the space. Resources added to the space are only visible within the space (and all spaces within it, as discussed next). In a transparent space, comments, tags, and the posted resources "percolate" down to the resource space. Non-members cannot see whats in the space, but can see the tags and comments on the resources.

- $C = \{RST, RLT, ENT\}$ denotes the configuration of the space. Because containers are used for a purpose, they typically include specific types of resources and relationships that acquire a particular meaning, and require a specific UI representation. For example, for instant communities the space will have panelists, attendees, presentations, questions, and the like as distinguished types, which in turn will be interpreted by the instant community UI. Specifically, a configuration is characterized by distinguished resource types RST, (e.g., papers, blogs, experiments), relationship types RLT (e.g., "next_version_of", "alternative_representation_of") and entity types ENT that take spe-

cific meaning inside this space (e.g., panel and questions in a space modeling panel discussions). They are characterized by listing the corresponding reserved terms and an informal description.

- Spaces can also follow a lifecycle defined by a particular design: for instance in a implementation of a KS modeling panel discussions the space will go through the phases involving - at least - the prior, during and post panel discussions. At each stage S in the lifecycle, the permissions and the way the UI renders the content may differ.

A KS is itself a resource, and as such KS can be included in other KSs, it can be annotated and linked as resources do.

4.3 Knowledge space platform and services

Kspaces are the platform and API on top of which KS applications (discussed next) are developed. In this section we describe the Kspaces platform and API to describe the services that are available to KS application developers.

4.3.1 Architecture

The high level architecture of the platform is described in Figure 4.2.

At the bottom we can see the Scientific Resource Space (SRS) Layer. This component implements the services that allow upper layers to manipulate the graph of scientific resources by adding, linking and annotating resources, and to select parts of the graph by defining a filtering criterion. It also connects to the distributed source of data through ad hoc adapters [5].

Above the SRS, the Knowledge Space Layer implements the KS primitives and exposes them as web services. It also implements the interface

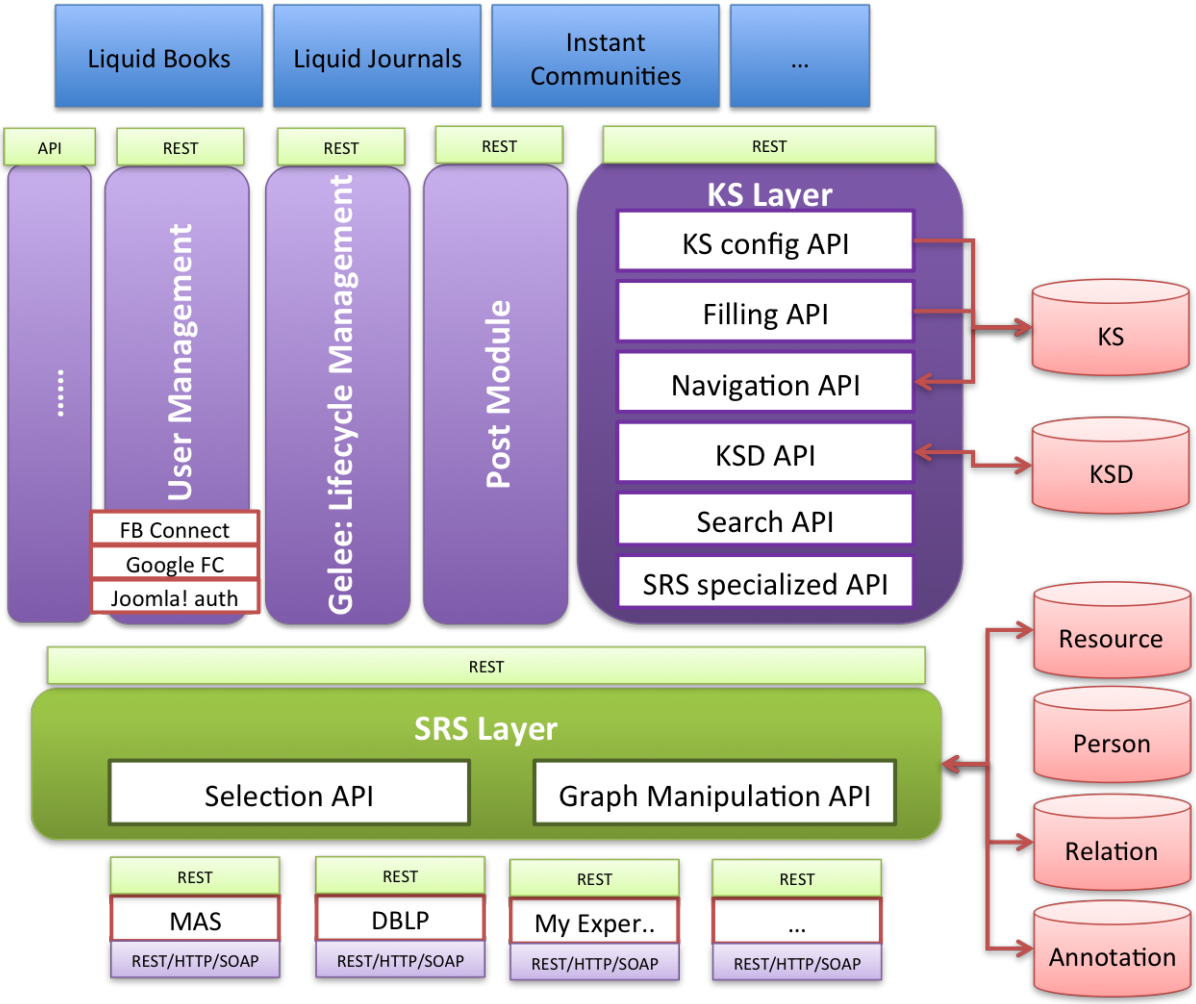


Figure 4.2: Architecture of the KS platform.

and the underlying support for registering KS applications. All the interactions are managed according to the context and application accessing the data, being this layer the one in charge of enforcing such rules and implementing the logic.

Along with the KS core component, a set of services that facilitates building designs is available: (i) the User Management provides the shared notion of user to the entire platform as well as the services for registering and authenticating users; (ii) the Gelee tool [6] provides services that facilitate modeling and managing and monitoring the lifecycle of spaces; (iii) the Post module provides a set of default services (e.g., link discovery, metadata extraction, hosting) that facilitates posting to a space in different environments. At the top we find the specific KS applications that exploit the advantages of the platform and provide interfaces and capture the knowledge of the scenarios they cover.

The KS module and supporting services are deployed on Amazon EC2, on a Glassfish Application Server and sitting on top an Oracle database. Once ongoing pilots are completed, releases of the hosted service are planned to be open to developers to add their own designs.

4.3.2 Collecting and posting knowledge

The post service is an essential part of Kspaces as they are in essence a way to collect and share knowledge. This requires the right tools and mechanisms for filling spaces with content in different environments. KS applications can then build on top of the primitives and provide the right tools for doing so.

In general information can be posted extensionally to Kspaces in the form of i) pointer to knowledge, such as URL of papers or datasets available online, ii) actual content (e.g., a pdf file or dataset), and iii) comments, tags, relationships, and other information that helps build a web of

scientific resources. When links or content is posted, Kspaces tries to find related metadata. For example, when a paper is posted, Kspaces extract metadata such as title and authors so that search can be facilitated. The posted item is also placed in the personal space of the poster, to be later archived, connected to other knowledge resources, copied to other spaces, shared, and the like.

In terms of technological means for posting, Kspaces provide the following services. These range of services are designed to minimize the posting effort thereby lowering the barrier to capturing knowledge.

- eMail service to capture the knowledge shared in research groups, where there is a strong email culture. People can send emails to colleagues and CC a Kspace (or send directly to Kspaces) with either links to URL or attachments.
- Mobile and tablet application for adding resources on the go, while reading a paper at a conference or in a train, by taking a picture of the paper and sending it to the system for the automatic recognition.
- Browser plugins to collect resources we find while browsing the Web. It allows users to feed the space using their search engine (e.g. Google Scholar) or publisher (e.g. Springer) of preference.
- Web interface for on site interactions. Each KS design provide its own UI metaphors and tools that allow researchers to interact with the space and to post scientific resources (by posting links or by drag and drop from the desktop)

4.3.3 KS Intentional Language

Another interesting way of collecting scientific resources is by defining the content intensionally. In this operation mode, space owners express the

"properties" of the content they want to include in the KS and the supporting platform takes the definition and continuously feed the space with the matching content. This requires a Domain-Specific Language (DSL) that exploits the characteristics of the KS model and the different purposes that the KS concept serves. From the user viewpoint, the intensional language properties are expressed via a UI (an example is shown in Figure 4.3). Conceptually, properties of resources of interest can be expressed in terms of resource type, such as blogs, papers, pre-prints, datasets, experiments, or whatever resource type is defined by the KS applications. Moreover, each type of content has its own set of attributes and particular relations. In defining filters on those attributes and relations, owners can focus the query on the properties they explicitly want in the scientific contributions. These properties are defined as a set of n-ary relations on the attributes (e.g., equals, not equals) and on the nodes of the participating relations, so it is possible to expand nodes and apply filters on them. Logical operators (e.g., conjunction, disjunction, negations) can then be used to connect filters and provide more complex filtering expressiveness. In the example interface we can see filters on the left side, on the input box at the top, and more complex filters implemented through navigation. To experiment the later, the user can click on an author and all the scientific resources authored by the person will be visualized on the same interface. This can be done for instance by expanding the "author_of" relation and applying a filter on it.

The rules we have mentioned can be also applied to sources. Thus, editors can reduce the scope and focus their attention on specific sources. For example, some users might want to get articles only from Springer (a source of certified scientific resources), others only from Google Scholar (a more comprehensive but noisy source). This is implemented in the example UI in Figure 4.3 as a filter on the left side of the workspace. In addition

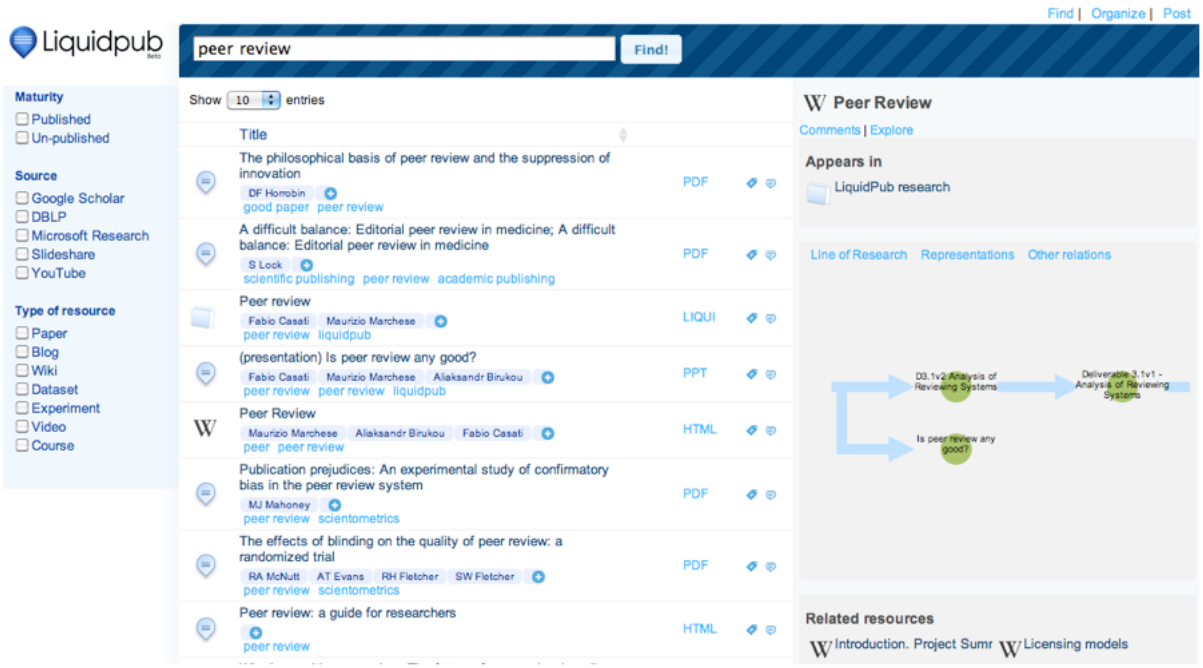


Figure 4.3: Example of Interface using KS intensional language.

to selecting items, an intensional expression can also include ordering, and grouping information and other properties. Ordering can be based on bibliometric indicators for published papers and on social metrics for all sorts of content (for information on the social metrics we refer the reader to [4]). In the future we plan to extend the capabilities of the intensional language with domain-specific operators, such as related resources, which in the current version are simply displayed (as shown in the figure, on the right) but cannot be used yet as part of the query defining the intensional expression.

The properties of the language are currently implemented on the interface of Figure 4.3 and are available for testing at http://journal.kspaces.net. The current implementation translates the interactions into REST calls to the backend, which in turn translates the requests into a set of SQL queries to our internal database.

4.4 Knowledge space Applications

Kspaces essentially are a general-purpose repository and API that can be used to develop applications for specific purposes around the area of collecting, linking, sharing, and finding scientific knowledge. For example, instant communities are a particular case of kspaces and can reuse the KS infrastructure and API as foundations. Specifically, they collect knowledge in spaces (the panel, or the specific topics of the panel), they allow to link, tag, and annotate knowledge, they require a post infrastructure that makes it easy to add content, they require the ability to create views (spaces over spaces) to extensionally or intenaionally specify subsets of content to be shared, and the like. They also have a lifecycle that dictates when users can post (at the start only panelists can add material, then all attendees can, in a post-oriented UI, and finally the community becomes open in a find and share oriented UI.

Instant communities also introduce specific entities, resources, and relationship types, and a specific UI that interprets the entity, resources and relationship types and presents information based on a specific interaction design. The UI is supported by an application-specific API that sits on top the KS API and limits/interprets the way it is used to fit the need of the instant communities UI. Sharing insights and material with colleagues during a panel session at a conference is for example different from sharing fragments of a textbook with co-authors. While the underlying model is similar and the APIs can be reused to a large extent, there will likely be concepts and UI interaction patterns that are specific to the task at hand. We argue that having a KS layer (and SRS model) that is useful to (and facilitates the realization of) many KS applications and having KS applications that are targeted at eliciting knowledge in various contexts can help reduce the barriers to knowledge sharing.

A KS application (KSA) is characterized by a conceptualization and by implementation artifacts that expose them in the way deemed appropriate for the target user and target purpose. Formally the conceptualization of a KS application is called a KS design

$$KSD = \{DESC, RST, RLT, ETT, SVC, LC\}$$

that includes:

- A name and informal description ($DESC$)
- A set RST of resource types which are meaningful in the context of the application. The semantic is defined informally and assumed to be communicated via the app UI or implicitly known. For underlying KS layer, the resource type is just a string. book chapter or panelist presentation are examples of resource types.
- A set RLT of relationship types, with the same consideration as above.
- A set ETT of entity types (e.g., panelists, panel moderator).
- A set SVC of KS services that are made avail to KS users. An example of service is post panelist presentation.
- A lifecycle LC that defines the states that a KSD can go through, and the allowed state transitions. For each state, only certain actions can be performed, by certain entity types. An example is the IC lifecycle described earlier.

The above fields go to configure the corresponding elements in a KS and are optional. In addition to the design, the KS application includes implementation artifacts that interact with the users on the one side and with the KS infrastructure on the other. These artifacts are the UI to interact with the app users, which will display the design-specific concepts

as desired and appropriate, an API that will implement the services, and possibly read-only database structures (such as materialized views) that may be required for performance reasons.

The API and UI of a KSA are essentially a different way to configure and expose KSs and the content in/access to the SRS. KSA can in fact only access the SRS via the KS API so they do not in fact add functionality, they only package it, expose it, and constrain it in a different way. For example, KSA services can be a restricted set with respect to KS services, or they may be a composition of multiple services. In this way we can "safely" add an arbitrary amount of apps as they will not negatively impact or render inconsistent the KS infrastructure, they only add different ways to consume it. As it is already begun to happen, the idea here is that the community develops and extends apps that provide novel ways to capture and share knowledge in specific contexts, where the KS API will then evolve by observing which are the common needs of several apps.

For example, the Instant Communities KSA considers some particular resource types such as panelist presentation, book chapter, along with some general ones such as paper, experiment, dataset, etc. As seen in the "panel page" of Figure 4.4 , panel presentations have particular treatment on the UI (preview on the right side), as required in this particular context. This KSA also assumes particular entity types that extend the basic model, such as panelist, panel moderator, question, etc. In the case of questions, they can be posted to kspaces in the same way resources can, but their nature, treatment and operations differ (questions can voted, ranked and then posed to panelists). Instant Communities also leverages specific relation types, to which it assigns an agreed semantic (and also graphical interaction patterns in the interface).

- Clustering relations denote topic relatedness. During a panel, people post content "near to" other existing content do denote that the infor-

mation is related. For example, an attendee may upload a document or comment over the presentation of one of the panelist to denote that the comment or posted item are related to the presentation. The instant community interface facilitates this.

- Temporal relations (such as next_version_of) model the evolution of a resource, be it a paper or dataset or anything else. This is a natural behavior of research dissemination where for example we write a preliminary version of a paper and then we extend or refine it. Or, we clean or add more data to a dataset.

- Structural relations represent arbitrary relationships between contributions, where the relationship is described by annotations. For example, a paper can be reporting on a dataset in that it describes results of experiments on that dataset.

- Representation relations allow us to model the multi-faceted aspect. For example, a paper can have associated slides and datasets, and so be deemed as a complex multi-faceted artifact, including artifacts that encode (part of) the same knowledge but have different representation.

The last three relation types are not meant for panel sessions and conferences, but for the personal space, where the user can organize its own content. We refer the reader to our earlier work for more details [4]. The Instant Communities tool is currently in production and available upon request.

4.5 Related Work

The last years have witnessed the rise of an increasing number of social bookmarking and annotation services in the scientific communities [27].

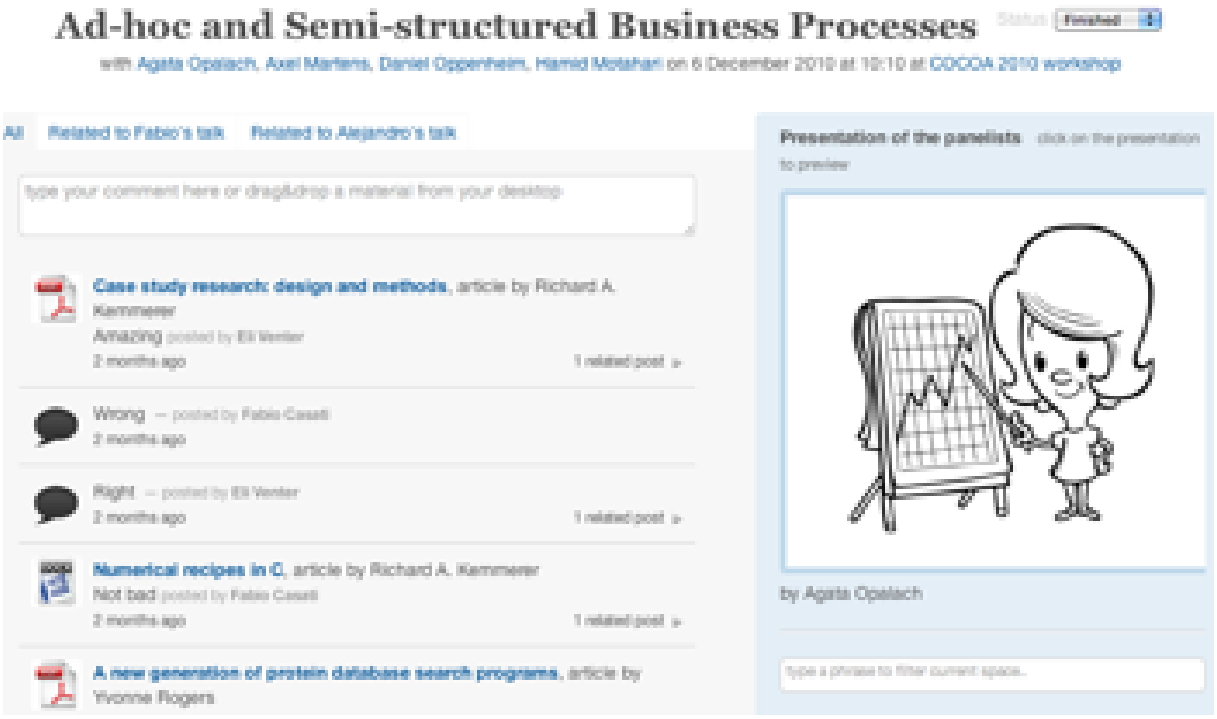


Figure 4.4: Instant Community Application UI.

Following the success of other popular but generic social bookmarking sites such as delicious.com Zotero, CiteULike , Connotea , Mendeley are examples of scientific social bookmarking services that focus on sharing and organizing academic references.

Zotero[1] is an extension for the Firefox browser that enables users to manage references directly from the Web browser. Users can bookmark publications, and then add their own personal tags and notes and in the current 2.0 version - share them within groups. Similarly, CiteULike[2] is a free online service to organize academic publications. It was the first Web-based social bookmarking tool designed specifically for the needs of scientists and scholars [33]. It allows users to bookmark or "tag" URIs with personal metadata using a Web browser; these bookmarks can then

[1]Zotero: `www.zotero.org`
[2]Citeulike: `http://www.citeulike.org`

be shared using simple URI links.

Connotea[3] is run by Nature Publishing Group and provides a similar set of features to CiteULike with some differences. Metadata can be extracted and shared from Connotea in a wider variety of formats than from CiteULike. Moreover there is an API that allows software engineers to build extra functionality around Connotea. Mendeley[4] is a desktop and web program for managing and sharing research papers, discovering research data and collaborating online [23]. With Mendeley is possible to store bibliographies using a more powerful desktop-based client that automatically extracts metadata from PDF files, but it can only do this where metadata is available in an simple and easy to extract format.

These tools and services deal with sharing and collecting materials targeting groups and individuals. However, they are of general purpose, and thus, their effective usage is limited to a reduced number of scenarios. As a result, their adoption implies changing the culture of the group or the way they communicate. In Knowledge spaces, we take a different approach and focus on formalizing and capturing the properties behind different collaborative research scenarios, managing the trade-off of general vs. specific, and while doing so leveraging search as well as providing the basic tools for allowing new metrics and novel services. From this perspective, the services available today could be seen as designs on top of our infrastructure.

There has been also some interesting work on fostering collaboration in scientific conferences to manage information overload (e.g., [15] [60]). These works point to the need of increasing participation in conferences, as an example of small communities affected by the inequality in collaborative systems [36]. These work, though focused on the scenario of conferences, provide insights in designing scenario-specific solutions. In kspaces we go

[3]Connotea: `http://www.connotea.org`
[4]Mendeley: `http://www.mendeley.com`

a step forward to provide a platform for enabling knowledge capturing in different scenarios and facilitating knowledge sharing and transfer across multiple scenarios.

Interesting ideas can be learned from other domains. Mail2Tag [35] explores the use of mail as tool for sharing and organizing news in environments where the email is the main communication channel. It brings organization by using conventional mail properties to introduce tagging. The system also reduces the problem of overload and provides mail digests generated based on automatically generated profile. This profile is built by looking at the tags of the mails sent to the user. Other experiences in capturing the implict knowledge in a community can be related to the Eureka project [12] form Xerox company, where experiences (tips) where collectively gathered by technicians to improve the problems found while fixing a product. These two are excellent examples of how by using the specifics of the domain can bring great benefits in terms of knowledge sharing.

As the different applications on top could also be seen as sources of information providing different interfaces but mapping them to a common model, we could relate KS with the concept of dataspaces [20]. The key difference and value of our approach relies in going deep in a vertical domain, which requires particular models, tools and architecture. Different models and standard formats for scientific artifacts have been proposed, trying enable the representation of metadata of papers and other artifacts, to enable integration and interoperability (e.g., [45], [52], [28]). In our approach, on the contrary, we identify the layers that separate the model of scientific contributions and define a domain-specific model (KS) that focus on the sharing, capturing knowledge and consumption of scientific content as well as extending the functionality to cope with different scenarios. We are not dealing with formats but with the challenges that the conceptual and practical challenges that the Web has brought.

Linked Data represents an important line of work on connecting data on the Web using standard web technologies [11]. We see the work on this community as complementary to KS concept, in that i) it can be a source of resources and relations to be used to feed kspaces, and ii) kspaces could also contribute with linked data to the Web. KS differs however in the focus, requiring particular models and platform to provide effective tools for capturing, sharing, organizing and assessing scientific content. Another important difference of our approach w.r.t to existing models and systems is that the underlying model was elaborated not only with the idea of providing a communication channel for scientist to share, but also with the goal of reducing the problem of information overload and providing new ways of assessing researchers [4].

4.6 Findings, Status and Next Steps

Kspaces have been developed in the context of an EU project and will now be taken over by a startup. They are the results of several attempts and failures at arriving at a model for capturing knowledge, which we initially tackled by trying to impose a specific knowledge collection mechanism (that is, a single, specific KS app). The finding during the years of work on this tool is that, besides a proper conceptual model, we need very domain-specific and targeted applications if we want to lower the barriers to knowledge sharing based on the principles described in the introduction. When we followed this approach, we saw that the kspace concept resonates with many stakeholders interested in different aspects of knowledge sharing and dissemination, from societies like IEEE and EAI to publishers like Springer, the EU, museum owners, and the like. Correspondingly, we are implementing and piloting a number of different KS applications. The first in line are the instant communities and liquid books pilots (liquid books

are a platform and contractual framework for collaborative and continuous editing of books). We also have implemented Liquid Journals[5] [4], a model and tool for scientific dissemination in the Web Era, and Liquid courses[6] [3], a model and integrated platform for knowledge transfer and sharing in the context of learning. A KS App for the MAA museum in Cambridge and UCLA is in preparation.

[5]Liquid journals videos: `http://www.youtube.com/view_play_list?p=3DFD404A84F456A8`
[6]Liquid courses video: `http://www.youtube.com/watch?v=yqBhQRFfinE`

Chapter 5

Resource Space Management Systems

Baez, M. and Casati, F. *Resource Space Management Sytems.*
Parra, C. and Baez, M. and Daniel, F. and Casati, F. and Marchese, M. and Cernuzzi, L. *Scientific Resource Space Management Systems.*

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.

5.1 Introduction

Liquidpub[1] is an EU project within the "future and emerging technologies" category whose goal is to capture the lessons learned and opportunities provided by the Web and open source, agile software development to develop concepts, models, metrics, and science support services for an efficient (for people), effective (for science), and sustainable (for publishers and the community) way of creating, disseminating, evaluating, and consuming scientific knowledge [17].

Novel services for science are a hot topic these days. From social bookmarking sites to online ranking of scientists, these services try to assist scientists in sharing content and assessing people and their scientific contributions. These services are however still very much anchored to a traditional notion of publication and are only scratching the surface of what can be done to help scientists collaborate for the greater good.

An example of services that Liquidpub intends to deliver is that of Liquid Journals1 (LJ), that redefines the traditional notion of journal which was born at a time where the paper was the only possible form of non-verbal knowledge dissemination, printing was the scarce resource, and therefore peer review and pre-publication filtering was necessary. Liquid journals are based on these notions i) separation of publication from inclusion in a journal: contributions are posted online (without any review) or published in traditional journals following a traditional process, and then they can be included in an arbitrarily high number of LJs. Each LJ decides policies and rules to determine if a contribution is included. Essentially, LJs are ways to aggregate all sort of available content based on what is interesting and relevant for its readers. This can be done via review, collaborative filtering, looking at journals of people we consider highly, etc; ii) Everybody (even

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

individuals) can create and run LJs; iii) Papers are not the only source of knowledge. Blogs, experiments, datasets, slides, comments/feedback and the like are valid and useful forms of dissemination, some of them having the additional benefits of allowing early dissemination and therefore better collaboration. Including feedback as a form of contribution has the effect that it is considered as part of what is evaluated from a scientist and therefore it encourages giving feedback, which is fundamental to the scientific creation process.

All is driven towards what the purpose of a journal should be: providing people with interesting content to read, minimizing the dissemination overhead, and maximizing the collaboration. Current journals are a particular case of LJs. In terms of web services, liquid journals require an infrastructure that allows defining LJs and fetching/filtering content from the web based on profiles, preferences, recommendations, policies, and so on. The effort in developing the liquid journals is on the definition of a query language capable of capturing the notions of interestingness and relevance, and on the development of the underlying query engine on top of scientific resources on the web, capable of merging results from various resource managers (e.g. search engines, social bookmarking services), filtering and grouping the results according to the query definition and to rank them according to their relevance.

Another service LP provides is research evaluation (also based on LJs, but not only). Evaluation is a necessary aspect of research, not only to filter contributions but also to help select people for hiring or promotion. In this respect, the LiquidPub project aim at developing scientific metrics that i) take into account the different aspects of the research activity: that of creating content, filtering content, proposing good ideas, setting up good experiments, and ii) encourage good behaviors (sharing content early, providing feedback, etc) and that not only look at what people have

done but that try to assess interest in what scientist will produce. Besides defining metrics, what we want to provide is a way to make it easy for scientists and evaluation agencies to define their own metrics. To this end, we need to provide services that allow programmatic access to scientific data and metadata – both traditional ones (Google scholar, citeseer, citeUlike, SpringerLink,..) and more novel ones (blogs, liquid journals,), that allows for sophisticated features such as author disambiguation or for comparing people of different communities and therefore having different scientific metrics (this is hard because it is hard to define what a community is), and that allow people to easily define and plug in their own metric which use data from their favorite sources.

5.2 Implications for Research Spaces Management Systems

Given the above, we need a common platform to access the various kinds of scientific resources available on the web, in a way that it easy (or at least easier) to develop services for scientists on top. For this, such a platform should provide programmatic access to scientific resources, 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., Google scholar). The large (and growing) amount of scientific web applications providing access to these resources makes it practically impossible to design a monolithic infrastructure that incorporates all of them. It is then required that such an infrastructure provides an extensibility facility that allows adding new services as needed.

We have also seen the need for a set of specific services in the examples above: services for extending the evaluation with user-defined metrics, primitives to manage author disambiguation, services for crawling various

scientific metadata sites (e.g., for citations), services for observing resource usage (to provide recommendations), etc. To support applications like LJs, we need support for query that understands concepts such as relevance or interestingness, we need to be able to collect user feedback or observe users actions if possible, and the like. We have also observed the need for a uniform conceptual model for scientific resources that is sufficiently general but also specific enough to be useful.

The previous observations led us to the design and development of a resource space management system (RSMS) for scientific resources. For this we borrow notions from the principles of Dataspaces [20] to apply it to a space of scientific resources. A resource is anything that has a URI, but the specific aspect is that RSMS is specifically focused on services to support knowledge dissemination. These resources are managed by potentially different service providers (e.g., Google Docs, Google scholar, ...). We refer to these service providers as resource managers. In a nutshell, the characteristics of the RSMS and for all the applications we build on top are:

- Homogenous programmatic access to scientific resources and web services regardless of how they are implemented as long as they are web accessible (via browser or rest/soap API).

- Universality, to cover the large set of scientific resources of various kinds of scientific resources as described above, not just papers.

- Collaborative Extensibility, to facilitate extensibility by the community where developers can just register scientific services. We bootstrapped the system with a few key access and crawling services, but the key is how to avoid overloading the system with hundred of adapters to access the different resource managers.

From the functional sides, the key is in understanding (and designing, implementing) which kind of actions are supported by the resource managers, which kind of horizontal services should be provided because they are useful to a large number of scientific services, and what is the underlying resource model to be exposed to the horizontal services as well as to the services to be developed on top.

5.3 Scientific Resource Space Management

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. We base our Scientific Resource Space Management in the formal definition introduced in Chapter 2, where we define the notions or scientific resource, relationships, annotation and other entities.

On this model, we characterize a series of services that a Scientific Resource Space should support. In Figure 5.1 we show the overall architecture of our platform, Karaku, including the following functional components:

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

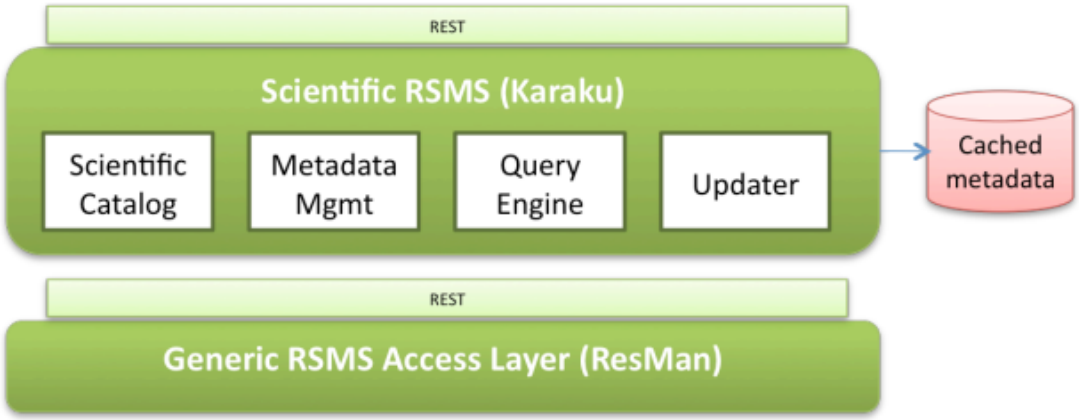


Figure 5.1: Architecture of the Karaku sRSMS

The scientific resource space would be useless without a welldesigned query language to take advantage of it.

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

- **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.

All these components provide a common model for the resources in the focused domain according to the model introduced in Chapter 2. 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 5.1 and described in the following.

5.4 Generic Resource Space Management

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 ac-

count 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.

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 how to bind scientific resources with actual resources, i.e., how to physically access resources distributed over different services.

5.4.1 Resource Space Model

In terms of models, RSMS is based on the notion of viewing every possible kind of scientific contribution available on the web as a scientific resource. Under this assumption, the web is a (scientific) resource space and the RSMS manages and simplifies access to these resources. *Resources* can be scientific contributions, people, and events, and can be grouped (communities are groups of people, proceedings are groups of papers, conference series are groups of events). Details can be seen in Figure 5.2.

Actions describe the services provided by resource managers and that allow us to operate with the resources (e.g., to share or search documents, or more complex actions such as crawling a web site for scientific metadata). On top of this we provide set of abstractions, to free upper layers of implementing resource specific operations.

At the level described above, the basic elements provide operations and properties, which are specific to the actual resource managers. For exam-

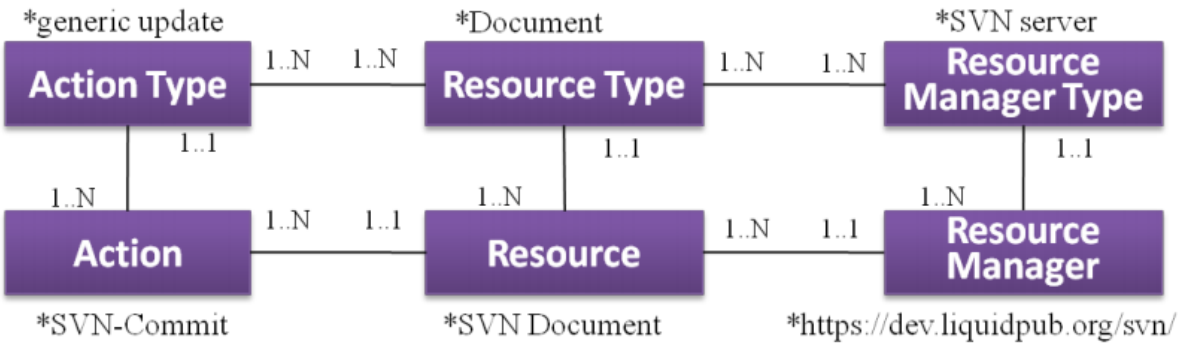


Figure 5.2: Resource Space model

ple, 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 *ac-*

Figure 5.3: Scientific Resource Space Architecture

tion 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

5.4.2 Architecture and services

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 5.3, 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* 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.

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.

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"

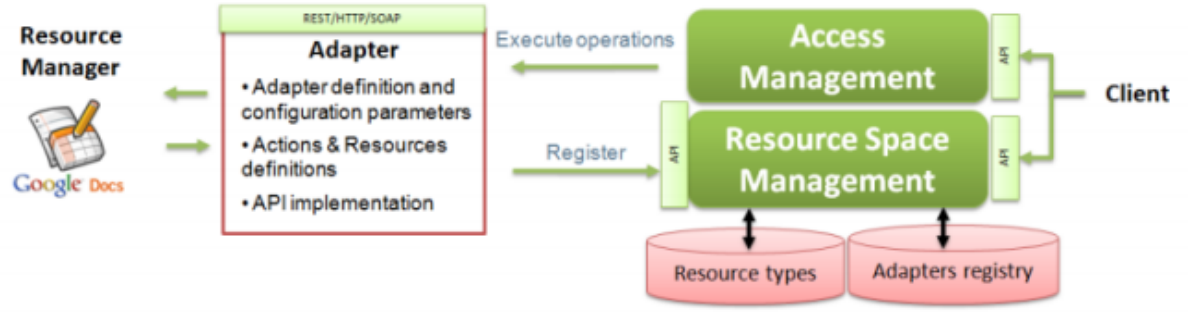


Figure 5.4: Adapter registration and operation execution

action type.

As illustrated in Figure 5.4, 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.

5.4.3 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 5.4 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.

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
<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>
```

`Location: 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`

```
<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>
```

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.

In the RSMS extensibility approach, the resource manager and the con-

cept 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.

5.5 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 [49] 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 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 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 tool [40]. 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.

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 5.5 we provide an example of how the users ideal journal is translated into a query.

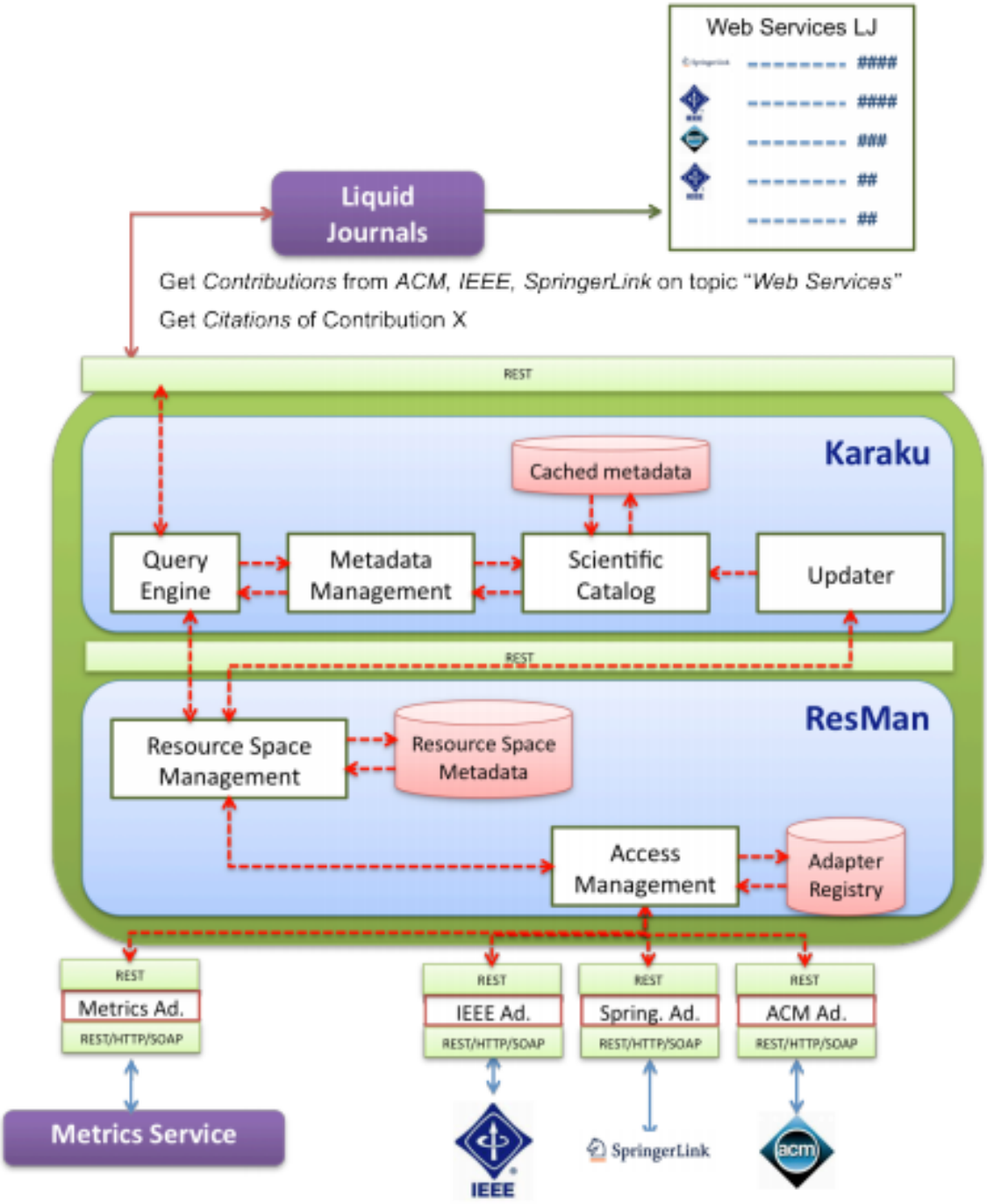


Figure 5.5: Adapter registration and operation execution

As seen in Figure 5.5, 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.

5.6 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 sourceindependent 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.

Chapter 6

Universal Resource Lifecycle Management

Baez, M. and Casati, F. and Marchese, M.

This paper presents a model and a tool that allows Web users to define, execute, and manage lifecycles for any artifact available on the Web. In the paper we show the need for lifecycle management of Web artifacts, and we show in particular why it is important that non-programmers are also able to do this. We then discuss why current models do not allow this, and we present a model and a system implementation that achieves lifecycle management for any URI-identifiable and accessible object. The most challenging parts of the work lie in the definition of a simple but universal model and system (and in particular in allowing universality and simplicity to coexist) and in the ability to hide from the lifecycle modeler the complexity intrinsic in having to access and manage a variety of resources, which differ in nature, in the operations that are allowed on them, and in the protocols and data formats required to access them.

6.1 Introduction

This work introduces concepts, methods, and a system for universal resource lifecycle management.

Nearly every artifact, from web pages, documents, wikis, code, to non-software resources (houses in construction, purchase orders, etc.) goes through a lifecycle. In a few cases, the lifecycle of these artifacts is supported by a tool that allows their modeling, automation, monitoring, and management. This typically happens when the lifecycle is formalized and strictly followed. For example, the process of approving purchase orders and procuring the goods is, in some large companies, supported by a workflow management system. In these cases, a system can interpret a formal definition of the lifecycle and execute/enforce it.

In the majority of cases however, the lifecycle is informally defined, and is executed, monitored, and managed "by hand", if at all. This is because generic process management tools are too complex and too rigid for this purpose, and are tightly coupled with the artifact they manage. For example, consider the execution of a software project, which includes the development and delivery of documents and code. The code is usually managed through a source control system, while the documents can be developed collaboratively online via the likes of Google Docs[1] or Zoho[2]. For each type of artifact, the team often defines a "quality plan" along with the lifecycle that the artifacts should follow. For example, design documents should be first reviewed and discussed by the development team, then reviewed by and discussed with the chief architect, and then signed off by the project manager. A unit manager, architect, or project manager, would like to know at a glance which documents are in a given status, which are late, and which have issues that need special attention. A team

[1] `http://docs.google.com`
[2] `http://www.zoho.com`

member/developer, would like to visualize the lifecycle of the documents he is in charge of, so that he knows what he is supposed to do with the document, and to automate the process of making it available to the team, sending it for review, collecting the reviews, sending it to the chief architect after revision, getting it signed off, and so on.

Today these types of lifecycles are modeled informally (sometimes even verbally) and they are mainly executed by hand typically by sending emails and editing access/visibility rights. The status is typically tracked by updating a MS project document or some spreadsheet.

Process and lifecycle management in these cases, using tools such as workflow managers, is unfeasible. First, the team would have to learn yet another tool, characterized by models (e.g., workflow models) typically fairly complex and anyways, despite marketing claims, targeted at programmers, not at users such as project managers. Second, the majority of everyday lifecycles are unstructured and flexible, and traditional workflow systems are not good at this (we will discuss this in detail in the related work section). Third, the progression through the lifecycle is often controlled by a human based on his /her judgment, not by an engine based on pre-defined rules. It is the developer, team leader, or project manager, who decides when the artifact can move to the next step of the lifecycle and which is this next step. Fourth, the decision of what to do at a given step in the lifecycle may itself change over time rather than being predetermined. For example, I may want to send the document to two rather than three reviewers, and decide who the reviewers are on the fly, or I may decide to post it and allow (i.e., set access rights so that) all my team to enter review comments. Fifth, in real projects typically there are a set of different kinds of artifacts (code, web pages, documents, etc) managed with different tools (CVSs, Web text editors, etc), distributed across the organization and managed by different owners. Using different lifecycle

management tool for each of these would be practically unthinkable.

This paper proposes an abstractions framework and a supporting environment that overcome these limitations and enable universal resource lifecycle management. We use the terms "universal" and "resource" as we want the system to manage whatever can be identified by an URI, regardless of its nature, managing application, owner, or location. We realize that such universality can often be at odds with ease of use, and indeed this is one of the challenges we face and address. The main characteristics of the proposed approach, also corresponding to the main contributions of the paper, are the following:

- The system is targeted at advanced web users (e.g., users comfortable with writing on wikis), not only programmers. The lifecycle model is very simple, essentially based on state machines. There are no complex features such as path conditions, transactions or exceptions.
- There is no need for modeling the resource being managed and its properties. The resource can be a black box from the lifecycle perspective. This is key both to universality and to keep the model simple from the perspective of the lifecycle designer who does not need to worry about the specifics of each resource.
- We support automation of operations on the resources (e.g., changing access rights, submitting for reviews, etc.), achieved by actions that can be associated to phases (states) and executed upon entering a phase. Actions are where both the complexity and the resource type-specific behavior reside (e.g., sending a Google doc for review also requires setting access rights, and the way this is done is Google Docs-specific). They are written by programmers, who populate a library of useful actions.
- The model is targeted at unstructured lifecycles, where there is a

high potential variability and the need to place the human in the drivers seat. For example, the lifecycle owner can determine when the resource transitions to the next phase or which is the next phase among the possible ones.

- The lifecycle management tool is hosted and available as a service, together with the lifecycle design interface and the monitoring interface, i.e., the interface a project manager would use to visualize status and history of the resources under her responsibility.

In the following we describe both the model and the prototypal system (named Gelee) in detail, together with the reasoning behind the various choices. We do this by starting from a concrete example (which is also the reason why we started developing this system), and extracting and abstracting requirements from it. Then, after discussing and comparing with the state of the art, we detail the model, the Gelee system architecture, implementation, and validation. We then discuss possible extensions and how these can be applied.

6.2 Motivating Scenario

6.2.1 EU Projects

At the heart of our interest in this problem was the participation in several European Union (EU) projects and in particular one in which we act as coordinators, called LiquidPub . So, we use this as a case study. EU projects involve people from different organizations working collaboratively (a project consortium) to achieve a project goal. EU projects are typically organized in work packages, each including tasks, deliverables, and milestones. Each of these has owners and collaborators (usually expressed as consortium partners, not people), and deadlines. In this motivating sce-

nario we focus on deliverables. A project has a number of deliverables ranging from 20 to 40 or more, depending on the size. In Liquidpub we have 35.

EU project coordinators typically define "quality plans" for deliverables, outlining essentially a desired lifecycle for them. This adds to the rules (and hence parts of the lifecycle) defined by the EU itself. For every deliverable there are one or more responsible parties playing different roles, with different levels of visibility or access rights. Moreover, each deliverable has its own lifecycle, which is comprised of different steps involving different activities and people.

For example, consider a typical scenario involving the production of a "State of the Art" deliverable. In the early phase of its elaboration, there is a small group of people sharing a document (maybe using Google Docs or a Wiki) in which they define the document structure and collaborate on specific sections, providing access rights as needed. Then, at some point (informally or formally defined as part of the quality plan) the document is shared with a wider group of people (specific reviewers, or the project team at large) to get feedbacks. The iteration of the elaboration and review phases continues until reviewers are satisfied. At this point the draft is transformed in the appropriate format, sent to the funding agency (EU in our case) for evaluation before a specified deadline, and possibly published on the project web site (either immediately or after EU approval). Very often, the work on the document continues, for example to prepare a survey paper for a journal. The above represents an ideal scenario. Internal deadlines can be missed, reviewers can be changed, phases can be shortened or skipped to make it in time, different deliverables can be merged into one or vice versa, etc.

6.2.2 Problem and Requirements Abstraction

From the above scenario we generalize requirements and desiderata for two classes of people involved in the project: project managers (who define the lifecycle, e.g., the project coordinator in our example) and artifact owners (who are responsible for driving the execution on an artifact, e.g., the responsible of a deliverable).

If we take the perspective of project managers people responsible for managing a relatively large set of artifacts - we would like to:

- Define the lifecycle of the different artifacts (we use the terms artifact and resource interchangeably). For example, define the quality plan that describe what every deliverable should go through. An example is given in Fig. 6.1.

- Associate the lifecycle to resources, possibly customizing it as needed for the resource (some deliverable may require specific treatment, for example our state of the art deliverable that was developed by integrating pieces done by the various project partners).

- Avoid as much as possible - the concerns of resource-specific details. We dont want to define different models based on whether the deliverable is done with Google Docs, or latex over Subversion.

- Monitor lifecycles. We (as project managers) would like to be able to have a picture of the status of the lifecycle for each artifact at any given point in time, with particular attention to delays.

- Simplicity. The user is the average scientist doing research, not a programmer. These kinds of users should be able to define, execute, and monitor the lifecycles.

- Flexibility and robustness. The web has taught us that things that work well are not only those that are simple but also those that are robust to failures or imprecision. Ideally it should be possible for the lifecycle to be partially specified and still be usable and useful for managing the artifacts evolution.

If we take instead the perspective of the artifact owner, we identify the following requirements:

- The owner should be able to go through the lifecycle, advancing from a phase to the next, and while doing so, (automatically) initiating and executing the necessary actions.
- The execution should be independent of the specifics of the resource. For all lifecycles, owners "simply" have to decide when they are ready to progress to the next phase.
- The abstraction and interfaces should be simple and integrated with the tool managing the resource, to simplify usage.
- The owner should have the possibility to deviate from the prescribed lifecycle. Changes (such as skipping a formal internal review due to delays) are the norm and imposing a fixed model would make the tool and abstractions useless. Furthermore, some parts of the lifecycle may be left to be decided by the owner or may have been unknown/undecided at lifecycle definition time. This means that the lifecycle for each object is only loosely defined beforehand.

Today, resource lifecycles in contexts like project executions are in the vast majority of cases managed by one tool: Microsoft Project. The reason is simple: MS Project is simple, intuitive, and imposes little or no unnecessary overhead. The challenge that is laid out for us therefore is to

provide a way to facilitate the definition and execution of lifecycles and the management of the various artifacts and their progress while achieving to the possible extent a level of simplicity, flexibility, and intuitiveness similar to that of MS Project.

6.3 Related Work

6.3.1 Workflow Management Systems

Workflow systems allow the definition, execution, and management of workflows. In general, workflow systems describe a business process as a set of tasks, to be executed in the order defined by the model. They are related to our work since they describe a flow model and actions to be executed on objects. They are however different since i) they do not focus on lifecycle management (they do not focus on the evolution of an object, but rather they model arbitrary actions to be executed by human or automated resources), ii) they are fairly rigid and prescriptive (they work well for structured, repeatable processes), iii) they are targeted to programmers and often designed for mission-critical applications (in fact they are not significantly less complex than Java for example), and iv) the corresponding software platform is large and complex to operate and maintain. Interesting lessons can however be learned by looking both at research in workflow evolution and adaptive workflow and at research on semi-structured workflow models, including in particular scientific workflows that are targeted at scientists.

In the area of adaptive workflows, several approaches have been proposed to provide dynamic process management [44][19][56], mostly focusing on managing migration of instances when the corresponding model is changed. In this paper we approach the problem by decoupling (or as we define later, light-coupling) instances and models, and automated migra-

tion is not required also because the progression of the flow is always done by humans.

A similar approach to the flexibility we offer in the lifecycle management is provided by the PROSYT system [18]. PROSYT takes the artifact-based approach in which operations and conditions for these operations can be defined over the concept of artifact type. Nonetheless, each artifact type defines just one possible lifecycle, and runtime lifecycle model changes are not allowed. This coupling reduces expressiveness and generality. In contrast, our approach provides independence from the resource being managed (universality), late binding of phases, actions, and resources, and we focus on simplicity in the model and system due to the nature of our target users.

With a different target, scientific workflows were developed for scientific problem-solving environments, in which experiments need to be conducted. Experiments can be considered as sets of actions operating on data, constituting possibly large data flows [58]. Due to the nature of the environment, it is not often possible to anticipate a scientific workflow, so model-changes and user intervention at runtime are necessaries to provide flexibility. Other requirements like reproducibility, detailed documenting and analysis are main concerns. Aside the fact that we take the artifact-oriented approach while this approach relies on a workflow, one main difference is that our model can be also descriptive. In other words, we consider important the monitoring also from the point of view of reflecting a step in the process, even if it does not involve a processing.

6.3.2 Document Management

The approach introduced in this work has roots also in the document engineering community. In this area, models and tools are developed around the concept of documents, which are particular types of resources. In [32]

the notion of document-centered collaboration is introduced. There, the activities of collaboration and coordination are considered aspects of the artifact rather than workflows. For this, they attach computation to documents (i.e. a word processor), whose actions define the workflow. However, this approach is focused in decoupling documents from workflows rather than providing a workflow modeling approach. In essence, this idea of separating the artifact from the workflow is aligned with our idea of decoupling artifacts from lifecycles, but we also build a flexible, reusable and simple lifecycle management model on top.

Flexibility is also important in this area. A framework for document-driven workflows was proposed in [57], which requires no explicit control flow. In this approach, the boundary of the flexibility is described by the dependency among documents, that is, one document being input of another. Nevertheless, as workflow operations are associated to changes in the documents, these changes must be done under the control of the workflow. In our approach, the lifecycle operations are associated to transitions, not to changes in the document. Thus, artifact processing (i.e., editing a Google Doc document) is freed from the model.

In [37], the processing of artifacts, from the creation to completion and archiving, is captured by lifecycles. Nonetheless, the flexibility offered is more focused on the artifact representation rather than lifecycle evolution and execution. Differing from this, our model provides flexibility in the lifecycle modeling and execution, and decoupling among lifecycle models and instances.

6.3.3 Lifecycle Modeling Notations

At present, there are a variety of models, notations, and languages for describing lifecycles. The most popular class of models is UML, and within UML the most common approach is to model lifecycles using state ma-

chines, that have exactly the purpose of modeling the state and evolution of an object, and the events that cause state transition [59]. State machines have been extended in a variety of ways, e.g., by allowing guards to be placed on transitions, to associate actions to transitions (statecharts [59]), and the like.

We essentially reuse finite state machines as the base for the lifecycle model we propose. The contributions of Gelee are not so much in the basic model, but rather in the instantiation and execution model, in the light-binding (described next) between models and instances and in how we cope with the heterogeneity of the possible resources to be managed and correspondingly with the different kinds of actions they support.

Other notations have been used to model lifecycles and processes. The most common ones are Petri nets and activity diagrams and their variations and extensions (which include also workflows and service composition notations such as BPMN [55]). We did not base our implementation on these notations as we find them more appropriate for describing workflows and procedures (generic sets of actions to be executed according to some ordering constraints) more than lifecycles (evolution of the state through which a resource goes through, and allowed actions in each state). In any case the essence of the differences of Gelee would still lie in the aspects mentioned above, not so much in the base notation.

6.4 Concepts, Models and Languages

In the following we first describe the lifecycle model at a high level, then discuss its execution semantics in terms of overall lifecycle executions and action executions.

6.4.1 Lifecycle model: basics

In essence, a resource lifecycle is a set of phases and phase transitions, similar to state machines and state charts. The phase describes the stage in life in which the resource is, while transitions denote possible evolutions. At any given moment, a resource is in one and only one phase. Figure 6.1 illustrates all the elements of the lifecycle with our example of Section 1.

At the lifecycle level, all the model needs to know of the resource is its URI and its type, a string whose main purpose is to denote which is the managing application. For example, resource types can be Wiki page, Google doc, Zoho project, SVN repository, etc. If the resource is password-protected, the model will also need login information. No other information is needed for the lifecycle to be able to manage the resource.

Phases can have associated actions. Actions are operations that are executed on the resource as the phase is entered. Examples of actions are: changing access rights, notifying reviewers, etc (see Figure 6.1). Actions have parameters which are typically instantiated as a lifecycle begins. For example, notify reviewers could have as parameter the reviewers list, which is an information we could have or not beforehand.

At the lifecycle model, neither the lifecycle composer (the one designing the lifecycle) nor the resource lifecycle owner (the person(s) in charge of advancing the lifecycle on a specific resource) needs to be concerned with how they are implemented. All actions associated to a phase are executed in parallel and anyway in a non-deterministic order. Any sequencing must be imposed either by splitting the phases. Actions are not guaranteed to succeed and there is no transactional semantic imposed by the model (nothing prevents the action itself, inside its implementation, of having a transactional behavior). The expected behavior is that when the actions complete, the lifecycle owner advances the lifecycle to the next phase

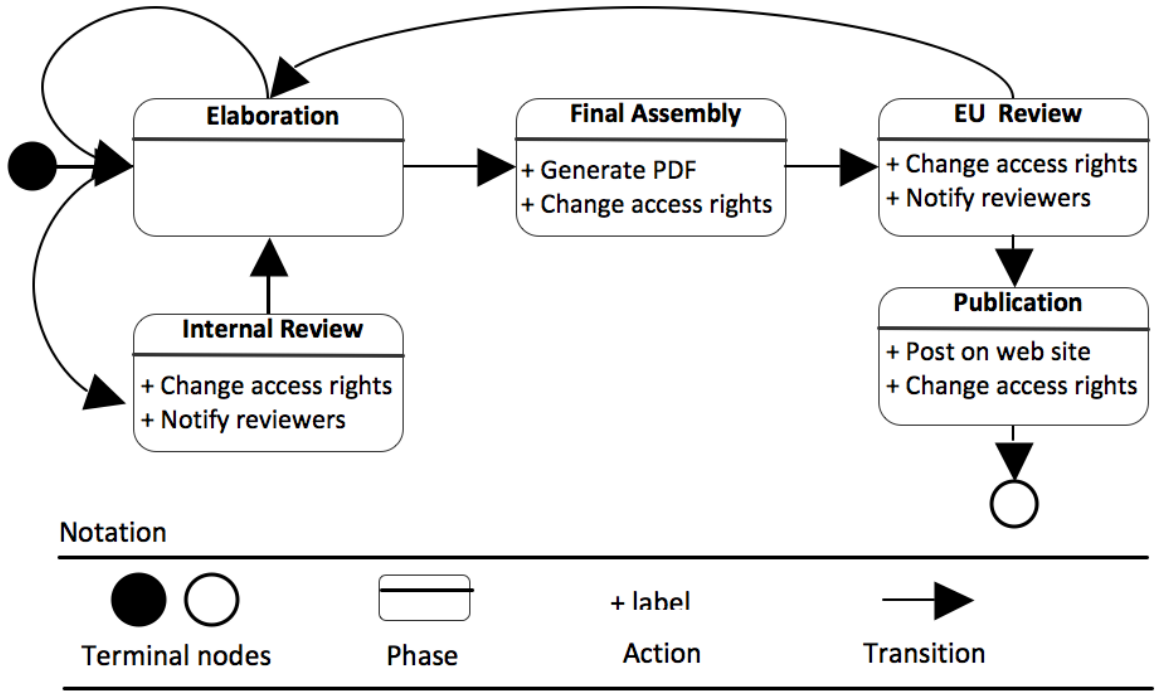


Figure 6.1: EU Project deliverable lifecycle.

(details on how this occurs are provided below).

When designing a lifecycle model, lifecycle composers can select the actions from a library (written by programmers). The actions they select will determine the resource types to which the lifecycle can be applied. Thus, models referring to resource-specific actions will have more limited applicability. Executing actions, however, is not the only purpose for having phases in a model. It is perfectly reasonable (and indeed useful) to have "empty" phases considering that one of the main purposes of lifecycles is also monitoring. For example, if the Elaboration phase in 6.1 involves editing a document in Google Docs we may still want to show that the current phase is "Elaboration", even if there is no action executed from the lifecycle. Finally, the model includes several other features not discussed in detail here, such as deadlines and time constraints as well as annotations. Annotations are in particular used to explain why a lifecycle owner does

not follow the standard flow, as discussed below.

6.4.2 Lifecycle Execution

A lifecycle instance is a particular execution of a lifecycle on a given resource. When the lifecycle instance begins, the lifecycle is associated to a specific resource and actions can be configured if necessary. The lifecycle remains active until an end phase is reached. End phases are phases with no associated actions, and their purpose is only to denote that the lifecycle instance is complete in a certain final state.

In Gelee there is no analogous of a workflow engine. The engine is the human, who executes the lifecycle instances (i.e., moves the tokens from phase to phase) and, while doing so, initiates the execution of actions. Another important aspect is that the model is descriptive rather than prescriptive. Its purpose is to describe a desired lifecycle (and the associated actions), not to impose it. In fact, the lifecycle owner can at any time move the token to any phase. One can argue that the model could include mandatory transitions or actions, but this is one of the many instances where we had to veto our desire to add features for the sake of keeping the model as lightweight as possible for lifecycle owners and designers.

Finally, owners can change the lifecycle followed by a resource, in other words they can change the model associated to a lifecycle instance.

The above denotes a light-coupling between models and instances. Owners can change the life of a resource without changing the model, and designers can change the model without affecting running instances if they so desire. If designers change a lifecycle model, they can request to propagate the change to running lifecycles. Upon receiving the request, lifecycle owners can accept or reject the change, and if they accept, they can state in which phase the lifecycle instance should end up in the modified model. Therefore, even in the presence of change, the problem of instance migra-

tions is here reduced to state migration. In terms of the lifecycle definition, the light-coupling between model and instance means that the XML that describes the lifecycle definition is self-contained.

A similar light-coupling exists between lifecycles and resources: nothing prevents several lifecycle to be defined on the same URI, and nothing prevents several lifecycle instances on the same URI to be running. Taking our example of 6.1 , in Table I we give an example of a lifecycle model definition using XML. This specification makes clear how the different components mentioned before are related.

Listing 6.1: Example of definition for a "Example fo an XML definition a the lifeycle"

```
<process uri=      >
<name>EU Project deliverable lifecycle</name>
<! Information about the version-->
<version_info>
 <version_number>1.0</version_number>
 <created_by>lpAdmin</created_by>
 <creation_date>08/07/2008</creation_date>
</version_info>
<! List of suggested resource_types-->
<resource>
 <resource_type>MediaWiki page</resource_type>
</resource>
<! Definition of the phases-->
<phases_list>
 <phase id= elaboration >
  <name>Elaboration</name>
 </phase>
 <phase id= internalreview >
  <name>Internal review</name>
  <! Actions to be executed -->
  <action_call>
   <action>
    <name>Change access rights</name>
    <uri>http://www.liquidpub.org/a/chr</uri>
     <parameters>
     <! Parameters to be specified at design-->
      <param id= paramID > value </param>
     </parameters>
   </action>
    ...
  </action_call>
 </phase>
 <phase id= finalassembly >
  <name>Final assembly</name>
  ...
 </phase>
```

```
  ...
</phases_list>
<! The list of suggested transitions-->
<transition_list>
 <transition>
  <from> BEGIN </from><to>elaboration</to>
 </transition>
  ...
</transition_list>
</process>
```

6.4.3 Actions

Entering a phase triggers the execution of the associated actions. The same compromise between definition and runtime flexibility that exists in the lifecycle model is provided to actions. The actions parameter can be fixed at definition time, instantiated at lifecycle instantiation time, or as the corresponding phase is entered. At execution time, the action is invoked by calling an URI that identifies a web service (either REST or SOAP), passing as parameters a link to the object and a callback URI.

Upon completion, or periodically during execution, the action can then call the callback URI and update on its status. The status messages are arbitrary except two defined by the model, corresponding to failure and successful completion. The status messages have only information purposes. Their interpretation or follow-up actions are left to the owner.

The attentive reader will have noticed that there is no analogous of workflow data, neither following the blackboard approach nor the data flow approach [2]. The owner inserts all parameters by hand. Any additional desired behavior must be part of the action implementation (as we discuss in the following).

Actions are associated to resource types, and represent operations that can be applied over the resource (also depending on what the native resource management application allows). For example, Google Docs service provides a REST API that allows us to perform operations over instances

of the spreadsheet type. Some of these actions are important for the point of view of the model, such as the ones that allow us to i) perform CRUD operations, ii) define access rights, and iii) subscribe to changes.

Notice that in this way the actions hide the specificities of each resource type. Indeed, it is also possible to define the same lifecycle and the same actions on resources at different types (e.g. Google Docs and Zoho for documents, Picasa and Flickr for photo albums, and control version systems such as CVS or SVN). This is done by mapping the same action name to different action implementations based on the resource types. Details will be provided in the next section.

We mention here, that the proposed approach could have many interesting uses looking at the growing number of hosted services that provide access to heterogeneous artifacts. Thus, the possibility of handling external resource makes this approach an attractive base for integrating such objects with user-defined processes.

6.4.4 Roles and Access Rights

During the lifecycle modeling and evolution, people are playing different roles. These roles define the set of operations users can perform over the lifecycle. In particular there are main roles: the lifecycle manager, the lifecycle instance owner and the token owner. The lifecycle manager is the person in charge of administrating a lifecycle, and thus, this role allows the user to design and modify the lifecycle. The lifecycle instance owner, however, is assigned to the person who instantiates the lifecycle on the resource. This role allows the user to drive and modify the lifecycle instance. Finally, the token owner role belongs to the user in charge of performing a transition at a given phase. Unlike the instance owner, its responsibilities are limited to follow the allowed transitions, and typically to specific transitions only. From the point of view of the resource we have

also the resource owner, as the person who has full access rights over a given resource, and who can assign permissions for it.

Thus, instance and resource owners can assign permissions or visibility rules over the instance and resources respectively. Nonetheless, access rules over the resource are performed by the platform that provides the resource, while lifecycle-related permissions are supported by the model.

6.5 Gelee at Work

This section describes the basic elements that allow the prototypal Gelee system to support lifecycle management.

6.5.1 Overall Architecture

The Gelee architecture is simple, especially due to the fact that there is no analogous of a workflow engine that progresses the flow from step to step. In essence, the system supports design and monitoring as well as invocation of actions that, from the core system perspective, are black boxes and are embedded into resource type-specific plug-ins that can be added as needed. As the primary goal of Gelee is to manage online resources and to have a system that is simple and usable, it was natural to provide lifecycle management as a service, and therefore hosted. Figure 6.2 depicts the high-level architecture, composed essentially of three layers: the data tier, the kernel and the user interface.

At the bottom of the figure we have the data tier, which includes the repositories for users and roles, resources and actions definitions, templates, as well as execution logs (including model evolution). The lifecycle manager is the hearth of the system, and it has a design time and a runtime module. The design time interacts with a lifecycle designer GUI (discussed next) via a SOAP and REST interface and receives definitions and modifications

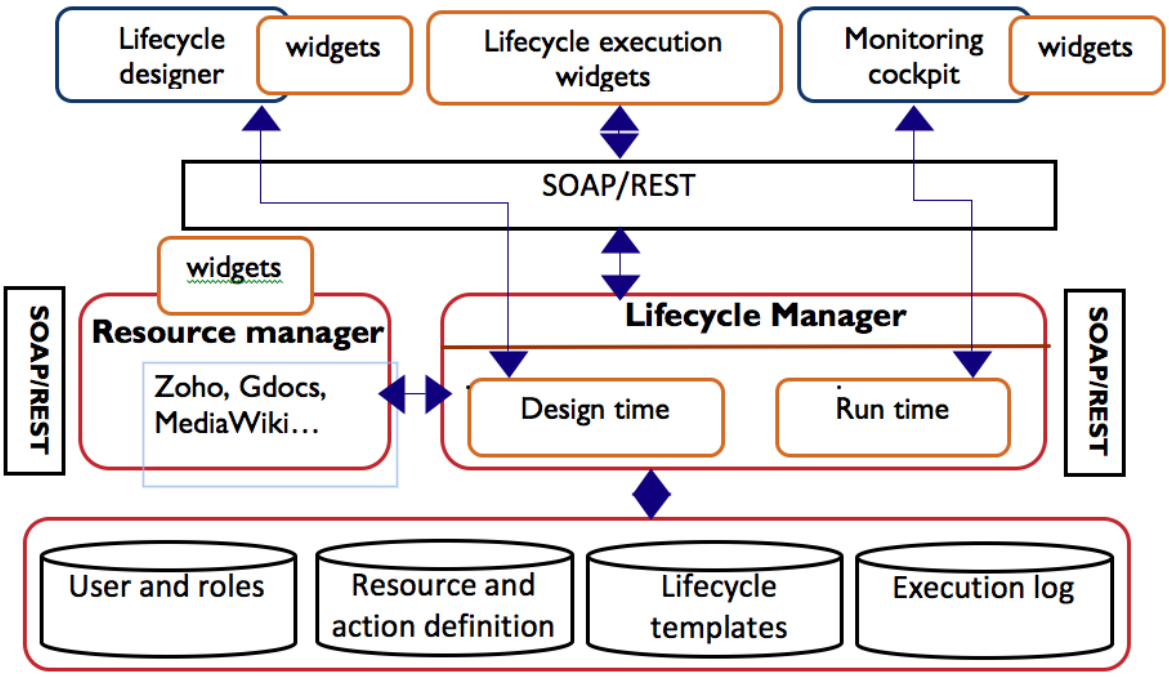


Figure 6.2: Gelee high-level architecture.

to a lifecycles. The runtime module receives lifecycle instance events (progression from phase to phase as dictated by the instance owner), sent by the lifecycle execution widgets, and action execution results, sent by resource plug-ins and discussed next. The interaction also in this case occurs via SOAP or REST messages. As a consequence of instance progression events, the lifecycle manager looks up the action list for the new phase reached by the lifecycle and contacts the resource type-specific plug-in to execute them.

6.5.2 Resources and Actions

Different resources are in general managed by different applications (Wiki, Flickr, etc..). In many cases, although 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 I can have the possibility of

changing the access rights, or sending it for review, or generating a PDF. Some of these actions are semantically equivalent but may require different parameters (i.e., the "signature" details are different). The implementation instead is certainly different and depends on the managing application.

This separation between action types and action implementations is another way in which Gelee supports light-coupling. Designers can define lifecycles (including definition of actions) that can be made applicable to different resource types. When a lifecycle is instantiated on a specific URI (and therefore on a specific resource of a specific type), actions types are resolved to specific action signatures and implementations.

The interfacing between the Gelee platform and a specific resource occurs through plug-ins or adapters. Developers can create adapters for any kind of resource, and implement actions that support a given functionality. The action implementation may correspond to an existing action type defined earlier for other resource types (e.g., send for review) or it can be a new action type that does not exist in Gelee. In both cases, the adapter needs to register the new action implementation with Gelee, to make Gelee aware that there is an action implementation for a specific resource type has been added, or that a completely new action type is introduced. The registration also includes information that Gelee needs for invoking the action.

The action definition is standard and includes information about i) the action type, which defines how to access the action; ii) parameters, and the time at which their values have to be associated; and iii) general metadata. This definition allows Gelee platform to handle all actions in a standardized fashion.

6.6 Conclusion and Future Work

In this paper we have described a universal resource lifecycle management model and the Gelee prototypal system . The current status of the framework is that components have been implemented (but not integrated) except the monitoring interface that has been only designed. Hence the source is available but the integrated platform is not yet available. Resource plug-ins currently include Google Docs and MediaWiki.

We tried to design the Gelee platform based on a very concrete case (i.e. European Projects) and based on what we and the people in the projects would like and would feel comfortable. In this kind of design and developments, we have the unique advantage that we ourselves ("we" writing the paper, "we" members of the project, but also "we" as researchers in general) are the users of the work and therefore it is easier to define users requirements users are comfortable with, especially in terms of resisting the temptation to make the approach feature-rich but then inflexible or complex. In this sense, the hardest parts of the work were in identifying the level of complexity of the model and the light-binding approach. The philosophy behind the design choice is to seek simplicity whenever we can and tackle complexity only if and when needed. Users who need simple things need not be bothered with complexity.

Other innovative aspects of our framework are (1) the action-resource model, which we believe provides a useful abstraction from the composition perspective; (2) extensibility and breadth of resource access and functionality. This is a significant departure for example from workflow models or even from service composition models.

The approach we have followed here is to put the complexity in the implementation of the actions, while keeping the model both general and simple, from the action perspective, to the composition designer. Indeed

the lifecycle model can be described in about a page and learned in a matter of minutes, literally. And it can be used to control any resource for which there is a plug-in.

The approach is also kept clean and extensible by leveraging plug-ins for resources, which can be externally managed and for which we only need a URI of the manager and an action interface for which we define the format, and that is very extensible.

In terms of future work, besides completing the monitoring aspect, interesting aspects include the integration with engines for those cases where engines are actually needed, and the challenge here lies in doing so keeping the same level of simplicity and flexibility. Another aspect we think it is interesting to explore is to link the lifecycle to complex resource types, and specifically to composed resources. This is a need we also have in the project, as sometimes the artifact (which in Liquidpub are called scientific knowledge objects) are structured, for example the state of the art is composed of the main documents, the references, presentations, etc., and managing a complex resource with components and with potentially independent but somehow interacting lifecycles is something that is part of our future explorations.

Chapter 7

Lessons Learned

In this work, we have addressed the limitations of the current model of *scientific knowledge dissemination* in the Web era. We have discussed the current problem of information overload as a result of other fundamental problems in the model of dissemination and proposed concepts, models and an infrastructure to reduce their effects. In this final chapter we summarize our findings and the legacy of this research project.

7.1 Impact on knowledge dissemination

The contributions of this research has been applied and adopted at different levels by the community. The tools developed have supported many important events and have been adopted by research groups as instruments for disseminatation. Concepts developed has been also adopted by the industry and raised discussions that captured the attention of specialized press.

7.1.1 Liquid journals

The Liquid Journal[1] model has been developed in cooperation with Springer and other partners of the Liquidpub project and has been piloted as part of ICST - and, as such, made available to a large community of users. The model enables information filtering through the pillars of a structured (but flexible) model for contributions, a model for journals that exploits the selection capabilities from the community, a consequent metric model, and finally a community discovery approach that identifies scientific communities, maps contributions to communities, and can therefore "suggest" contributions from different communities. Together this can address the information overload problem in science while maintaining the potential that derives from having a lot of information available, that of leveraging breadth in the search. Screenshots of our prototype can be seen in Figure 7.1.

These concepts and ideas motivated discussion in the scientific community and specialized press (Figure 7.3), who have seen the value and the need for a transition. Moreover, societies such as Complex Systems[2] welcomed the idea and started to publish their own liquid journal `http://www.complexssociety.eu/liquid_journal_of_complex_systems.html`.

7.1.2 Instant Communities

Instant Communities has served as platform to support the EU Commission flagship event for future and emerging technologies, fet11.eu. This event provided the first conceptual validation, where the IC concepts and applications have been found useful by the conference committee to invest resources to help integrate it and to advertise it to over a thousand attendees as a tool to support the event. In a survey after the event, the

[1] Liquid journals videos: `http://www.youtube.com/view_play_list?p=3DFD404A84F456A8`

[2] `http://www.complexssociety.eu/`

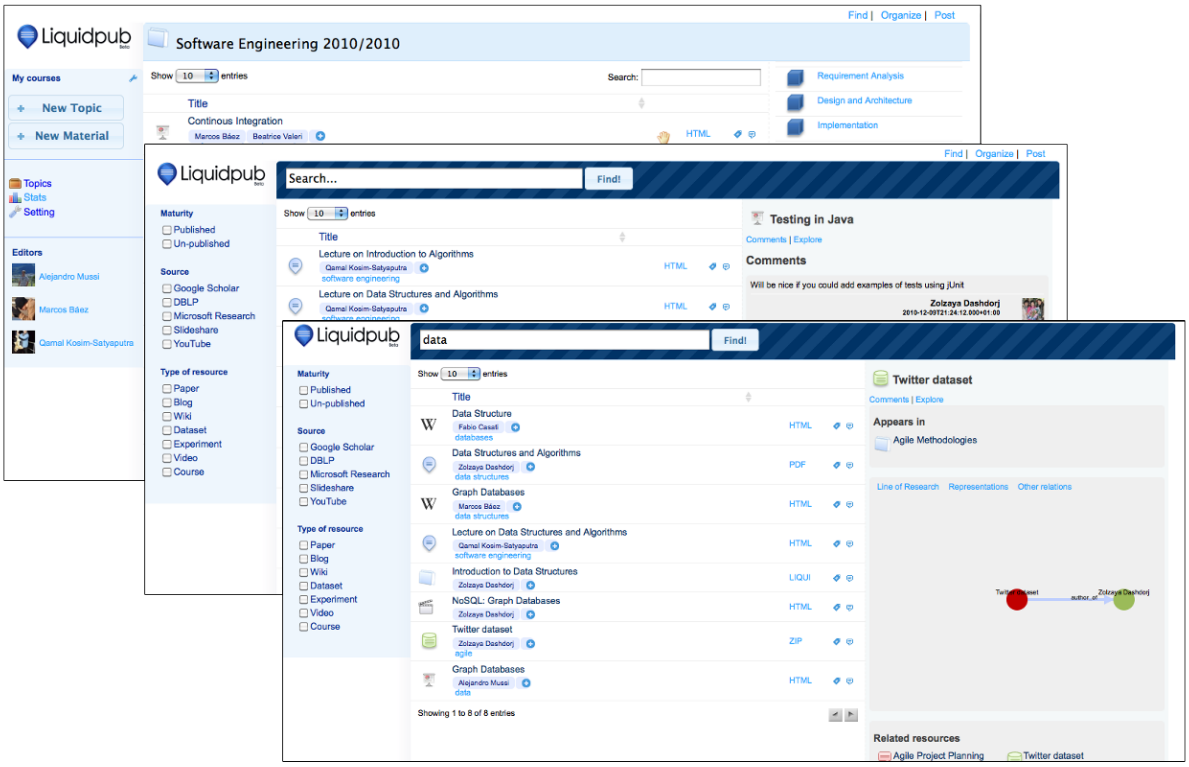


Figure 7.1: Liquid Journals sreenshots

Figure 7.2: Liquid Journals on MIT Techonology Review and BMJ (British Medical Journal) Blog, and other publishing and library blogs

tool recived mostly positive comments from people who use it. Still, only 30% of the people used the tool, the main reason stated in the comments, and confirmed by our observations during the event, was that they were not aware of the tool. The lesson learned from this experience was that, besides the technological benefits, the tool should have better support by the session chairs and moderators in order to be adopted.

The tool has also supported other events in last years, such as EUD4Services 2011, ComposableWeb 2012 and MDWE 2012. Instant Communities has provided actual support to these events, with the main motivation being the need for the service and not the evaluation of the tool. The feedback in all cases has been positive and led to requests for support in future events. At the time of this writing, the tool is providing regular support to the seminar series of two reasearch groups from University of Trento (Social Informatics and Big Data).

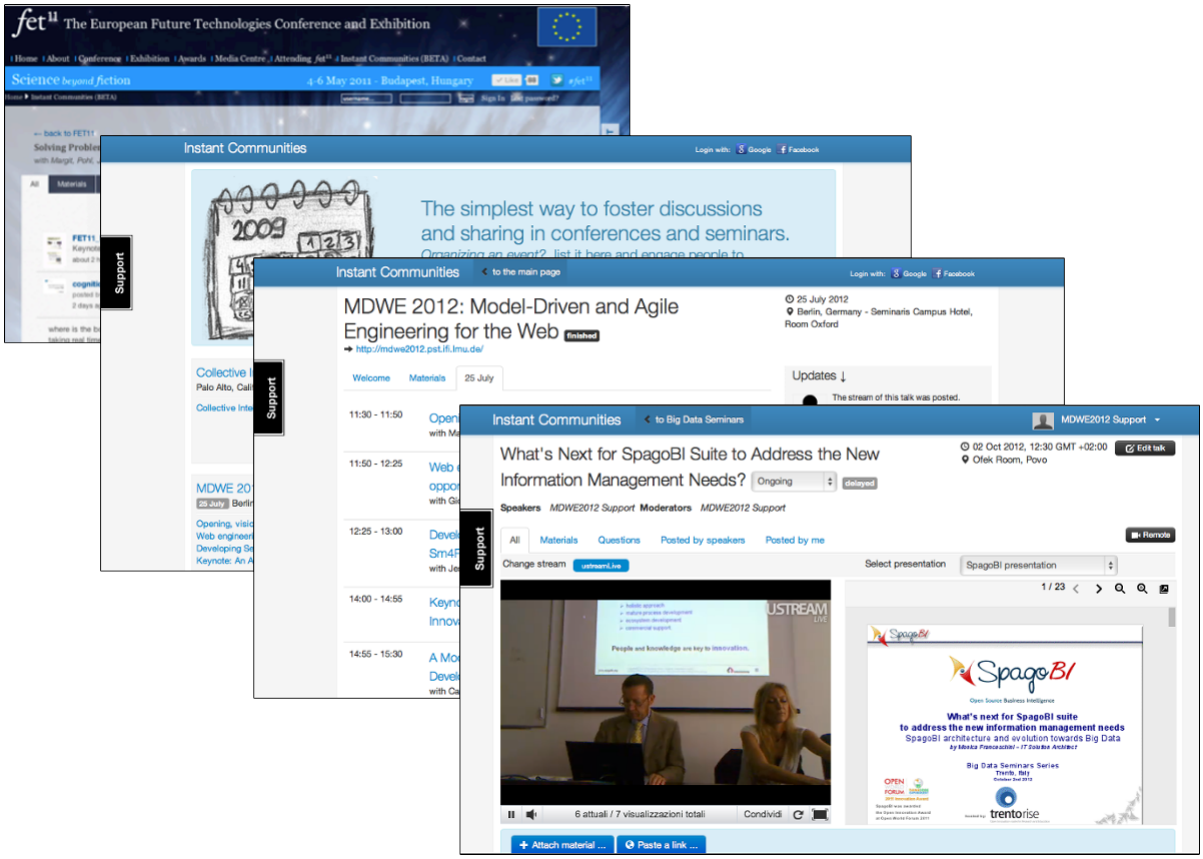


Figure 7.3: Instant Community at various events

Today, the tool is available at `http://ic.kspaces.net` and open to the community to support conferences and seminars. It is maintained by the Lifeparticipation group[3], which provides infrastructural support to keep it online. There is a branch of the tool designed for courses, namely Streamscience[4], that has been providing support to several university courses around the world.

7.1.3 Knowledge Spaces Platform

Kspaces is the result of several attempts and failures at arriving at a model for capturing knowledge, which we initially tackled by trying to impose a specific knowledge collection mechanism (that is, a single, specific KS app). The finding during the years of work on this tool is that, besides a proper conceptual model, we need very domain-specific and targeted applications if we want to lower the barriers to knowledge sharing based on the principles described in the introduction. When we followed this approach, we saw that the kspace concept resonates with many stakeholders interested in different aspects of knowledge sharing and dissemination, from societies like IEEE and EAI to publishers like Springer, the EU, museum owners, and the like. Correspondingly, we piloted and implemented a number of different KS applications, from Liquid Journals, to Instant Comumunities, Streamscience, and others. These experiences has been documented in this thesis and expensivenly discussed in [29].

The key for this ecosystem of tools is the strong technological foundation. We have designed and implemented an abstraction layer for scientific services that leverage our notion of "scientific contribution" to provide seamless access to resources ditributed among different services over the internet [5] [39]. We also provided a model and implemented a prototype

[3] `http://lifeparcipation.org`
[4] `http://streamscience.org`

for flexible and lightweight processes, in order to cover the requirements of the spectrum of scenarios we targeted at. Although parts of this could not be used entirely given the maturity of the tool, the concepts were incorporated in the model and implemented in the platform. Finally we also explored how to facilitate building scenario-specific applications on top of our infrastructure using domain-specific mashups [50].

7.2 Final Remarks

In summary, in this thesis contribute concepts, models and infrastrurcture for scientific knowledge dissemination to reduce information overload effects. To this end, we enabled an ecosystem of knowledge capturing applications and a way to share and search for knowledge based on an understanding of community practices. We see this as a way to foster a byte-sized exchange of knowledge in all of its forms that can support an "agile" form of knowledge creation and dissemination, to complement the traditional scientific paper + peer review model which will continue to thrive and which is very good for exchanging reports of baked ideas and results.

The contributions of this work are currently going beyond the boundaries of scientific knowledge dissemination, to be applied in domains such as collective intelligence [8] and experience sharing [54].

Bibliography

[1] N. Agarwal, E. Haque, H. Liu, and L. Parsons. Research paper recommender systems: A subspace clustering approach. *Advances in Web-Age Information Management*, pages 475–491, 2005.

[2] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web services: concepts, architectures and applications*. Springer, 2003.

[3] M. Baez, B. Benatallah, F. Casati, V. Chhieng, A. Mussi, and Q. Satyaputra. Liquid course artifacts software platform. *Service-Oriented Computing*, pages 719–721, 2010.

[4] M. Baez, A. Birukou, F. Casati, and M. Marchese. Addressing information overload in the scientific community. *Internet Computing, IEEE*, 14(6):31–38, 2010.

[5] M. Baez and F. Casati. Resource Space Management Systems. In *Proceedings of the European Conference on Web Services (ECOWS 2009)*.

[6] M. Báez, F. Casati, and M. Marchese. Universal Resource Lifecycle Management. In *Proceedings ICDE*.

[7] M. Baez, F. Casati, and M. Marchese. Sharing scientific knowledge with knowledge spaces. *Social Informatics*, pages 308–311, 2011.

[8] M. Baez and G. Convertino. Innovation cockpit: a dashboard for facilitators in idea management. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion*, pages 47–48. ACM, 2012.

[9] M. Baez, D. Mirylenka, and C. Parra. Understanding and supporting search for scholarly knowledge. *ECSS 2011, 7th European Computer Science Summit*, 2011.

[10] B. Benatallah, F. Casati, D. Grigori, H. Nezhad, and F. Toumani. Developing adapters for web services integration. In *Advanced Information Systems Engineering*, pages 415–429. Springer, 2005.

[11] C. Bizer, T. Heath, and T. Berners-Lee. Linked data-the story so far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22, 2009.

[12] D.G. Bobrow and J. Whalen. Community knowledge sharing in practice: the eureka story. *Reflections*, 4(2):47–59, 2002.

[13] T. Bogers and A. Van den Bosch. Recommending scientific articles using citeulike. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 287–290. ACM, 2008.

[14] A. Broder. A taxonomy of web search. In *ACM Sigir forum*, volume 36, pages 3–10. ACM, 2002.

[15] P. Brusilovsky, D. Parra, S. Sahebi, and C. Wongchokprasitti. Collaborative information finding in smaller communities: The case of research talks. In *Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on*, pages 1–10. IEEE, 2010.

[16] D. Carmel, N. Zwerdling, I. Guy, S. Ofek-Koifman, N. Har'el, I. Ronen, E. Uziel, S. Yogev, and S. Chernov. Personalized social search based on the user's social network. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1227–1236. ACM, 2009.

[17] F. Casati, F. Giunchiglia, and M. Marchese. Liquid publications: Scientific publications meet the web. 2007.

[18] G. Cugola. Tolerating deviations in process support systems via flexible enactment of process models. *Software Engineering, IEEE Transactions on*, 24(11):982–1001, 1998.

[19] P. Dadam, M. Reichert, S. Rinderle, M. Jurisch, H. Acker, K. Göser, U. Kreher, and M. Lauer. Towards truly flexible and adaptive process-aware information systems. *Information Systems and e-Business Technologies*, pages 72–83, 2008.

[20] M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *ACM Sigmod Record*, 34(4):33, 2005.

[21] M. Hadjieleftheriou and V. J. Special issue on result diversity. volume 31(4), 2009.

[22] A. Haque and P. Ginsparg. Positional effects on citation and readership in arXiv. *Journal of the American Society for Information Science and Technology*, 60(11), 2009.

[23] V. Henning and J. Reichelt. Mendeley-a last. fm for research? In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 327–328. IEEE, 2008.

[24] P. Heymann, G. Koutrika, and H. Garcia-Molina. Can social bookmarking improve web search? In *Proceedings of the international conference on Web search and web data mining*, pages 195–206. ACM, 2008.

[25] JE Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences of the United States of America*, 102(46):16569, 2005.

[26] B.A. Huberman, D.M. Romero, and F. Wu. Crowdsourcing, attention and productivity. *Journal of Information Science*, page 0165551509346786v1, 2009.

[27] D. Hull, S.R. Pettifer, and D.B. Kell. Defrosting the digital library: bibliographic tools for the next generation web. *PLoS computational biology*, 4(10):e1000204, 2008.

[28] J. Hunter. Scientific models: a user-oriented approach to the integration of scientific data and digital libraries. In *VALA2006*, pages 1–16, 2006.

[29] Instant communities. demonstrator description, 2011.

[30] D. Kelly and J. Teevan. Implicit feedback for inferring user preference: a bibliography. In *ACM SIGIR Forum*, volume 37, pages 18–28. ACM, 2003.

[31] M. Krapivin, M. Marchese, and F. Casati. Exploring and Understanding Citation-based Scientific Metrics. 2008.

[32] Anthony LaMarca, W. Keith Edwards, Paul Dourish, John Lamping, Ian Smith, and Jim Thornton. Taking the work out of workflow: mechanisms for document-centered collaboration. In *Proceedings of the sixth conference on European Conference on Computer Supported*

Cooperative Work, ECSCW'99, pages 1–20, Norwell, MA, USA, 1999. Kluwer Academic Publishers.

[33] D.H. Lee and P. Brusilovsky. Social networks and interest similarity: the case of citeulike. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 151–156. ACM, 2010.

[34] N. Lossau. Search engine technology and digital libraries. *D-Lib Magazine*, 10(6), 2004.

[35] L. Nelson, L. Nairn, and E.H. Chi. Mail2tag: Lightweight information sharing services integrated with email. *Software Demonstration. ACM CSCW*, pages 6–10, 2010.

[36] J. Nielsen. Participation inequality: Encouraging more users to contribute. *Jakob Nielsens alertbox*, 9:2006, 2006.

[37] A. Nigam and N.S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428–445, 2003.

[38] N. Osman, J. Sabater-Mir, C. Sierra, A.P. de Pinninck Bas, M. Imran, M. Marchese, and A. Ragone. Credit attribution for liquid publications. *Deliverable D4*, 1, 2010.

[39] C. Parra, M. Baez, F. Daniel, F. Casati, M. Marchese, and L. Cernuzzi. A scientific resource space management system. 2010.

[40] C. Parra, M. Imran, D. Mirylenka, F. Daniel, F. Casati, and M. Marchese. A scientific resource space for advanced research evaluation scenarios. In *19th Italian Symposium on Advanced Database Systems*, 2011.

[41] D. Parra and P. Brusilovsky. Evaluation of collaborative filtering algorithms for recommending articles on citeulike. In *Proceedings of the Workshop on Web*, volume 3. Citeseer, 2010.

[42] S. Pinfield. Journals and repositories: an evolving relationship? *Learned Publishing*, 22(3):165–75, 2009.

[43] P. Pirolli and S. Card. Information foraging. *Psychological review*, 106(4):643, 1999.

[44] M. Reichert and P. Dadam. Adept flexsupporting dynamic changes of workflows without losing control. *Journal of Intelligent Information Systems*, 10(2):93–129, 1998.

[45] M.A. Rodriguez, J. Bollen, and H. Van de Sompel. A practical ontology for the large-scale modeling of scholarly artifacts and their usage. *arXiv preprint arXiv:0708.1150*, 2007.

[46] D.E. Rose and D. Levinson. Understanding user goals in web search. In *Proceedings of the 13th international conference on World Wide Web*, pages 13–19. ACM, 2004.

[47] J. Sandweiss. Essay: the future of scientific publishing. *Physical review letters*, 102(19):190001, 2009.

[48] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. *The adaptive web*, pages 291–324, 2007.

[49] J.W.T. Smith. The deconstructed journal–a new model for academic publishing. *Learned Publishing*, 12:79–91, 1999.

[50] S. Soi and M. Baez. Domain-specific mashups: from all to all you need. In *Proceedings of the 10th international conference on Current trends in web engineering*, pages 384–395. Springer-Verlag, 2010.

[51] T. Strohman, W.B. Croft, and D. Jensen. Recommending citations for academic papers. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 705–706. ACM, 2007.

[52] S. Sufi and B. Mathews. Cclrc scientific metadata model: Version 2. *Final report. Council for the Central Laboratory of the Research Councils. Report No: DL-TR-2004-001*, 2004.

[53] J. Tang and J. Zhang. A discriminative approach to topic-based citation recommendation. *Advances in Knowledge Discovery and Data Mining*, pages 572–579, 2009.

[54] B. Valeri, M. Baez, and F. Casati. Comealong: Empowering experience-sharing through social networks. 2012.

[55] W. Van der Aalst, A. ter Hofstede, and M. Weske. Business process management: A survey. *Business Process Management*, pages 1019–1019, 2003.

[56] W.M.P. Van der Aalst, M. Weske, and D. Grünbauer. Case handling: a new paradigm for business process support. *Data & Knowledge Engineering*, 53(2):129–162, 2005.

[57] J. Wang and A. Kumar. A framework for document-driven workflow systems. *Business Process Management*, pages 285–301, 2005.

[58] M. Weske, G. Vossen, and C.B. Medeiros. *Scientific workflow management: WASA architecture and applications*. Citeseer, 1996.

[59] D. Wodtke and G. Weikum. A formal foundation for distributed workflow execution based on state charts. *Database TheoryICDT'97*, pages 230–246, 1997.

[60] C. Wongchokprasitti, P. Brusilovsky, and D. Parra-Santander. Conference navigator 2.0: community-based recommendation for academic conferences. 2010.

[61] S.A. Yahia, M. Benedikt, L.V.S. Lakshmanan, and J. Stoyanovich. Efficient network aware search in collaborative tagging sites. *Proceedings of the VLDB Endowment*, 1(1):710–721, 2008.