



D5.1v3

Design of the Liquid Publications Integrated Platform

Maintainer/Editor-in-chief	Aliaksandr Birukou
Core authors	Marcos Baez, Aliaksandr Birukou, Fabio Casati, Ronald Chenu, Muhammad Imran, Maurizio Marchese, Matus Medo, Daniil Mirylenka, Nardine Osman, Cristhian Parra
Maintainers/Editors	Marcos Baez, Ronald Chenu, Nardine Osman
Reviewers	Peep Kungas, Nardine Osman
LiquidPub research group leaders	Fabio Casati, Roberto Casati, Marlon Dumas, Ralf Gerstner, Fausto Giunchiglia, Maurizio Marchese, Gloria Origgi, Alessandro Rossi, Carles Sierra, Yi-Cheng Zhang
LiquidPub project leader	Fabio Casati

Grant agreement no.	213360
Project acronym	LiquidPublication
Version	v3.0
Date	April 27, 2011
State	Liquid
Distribution	Public

Disclaimer

The information in this document is subject to change without notice. Company or product names mentioned in this document may be trademarks or registered trademarks of their respective companies.

This document is part of a research project funded by the European Community as project number 213360 acronym LiquidPublication under THEME 3: FP7-ICT-2007-C FET OPEN. The full list of participants is available at <http://project.liquidpub.org/partners-and-contributors/liquidpub-teams>.

Abstract

This document presents the overall architecture of the Liquid Publication integrated platform and describes all components in detail. In addition to that, it also describes LiquidPub applications. This final version not only introduces advances in the development of the platform but also a more mature understanding of the requirements and possibilities, explored in the LiquidPub applications covering the aspects of knowledge creation, dissemination and evaluation.

Keyword list: **design, architecture, applications, platform, instant communities, liquid journals, liquid books, liquid conferences**

Executive Summary

The goals of LiquidPub are realized through concepts and models which are supported by an IT solution (the LiquidPub platform) and an ecosystem of services. This document describes the overall platform and the applications through which LiquidPub concepts can be exposed to the scientific community.

In a nutshell, the way in which the LiquidPub platform (Figure 2) supports the LP vision is the following:

- There are many different contexts in which *LiquidPub applications* can capture and disseminate knowledge, and the applications provide metaphors and interaction patterns for the different contexts in which one can capture and disseminate knowledge. For example, *liquid conferences* facilitate conversations on papers (or parts of a paper), while community-oriented applications like *instant communities* aim at capturing implicit knowledge about scientific contributions and sharing it within a community of interest. Within the LP platform, the applications are represented as the vertical rectangles in Figure 2.
- A *Scientific Resource Space Management System* (SRS) stores and manages SKOs and allows access to them. It includes
 - A SKO repository, that models information collected by the different applications, and also acts as an integrated repository of liquid knowledge.
 - A SKO management API, called Knowledge Spaces, that allows both to access SKOs as well as ways to group them and define access rights.
 - Extract Transform Load (ETL) routines and wrappers that connect LP with existing sources of information and in general with the non-LP world. An example of this consists in getting metadata about scientific publications from external sources such as CiteSeer, DBLP, Microsoft Academic Search (MAS) and ePrints via API or wrappers to the LP platform so that such information can be integrated into the SKO repository or the applications, or can be used by the reputation tools.
- A *trust and reputation* module that processes information from the SKO repository and from the applications to compute metrics that (i) reflect the collective opinions of scientists over scientific contributions, (ii) go beyond metrics with well-known flaws such as citation counts, and (iii) expose metrics that encourage behaviors that are beneficial for science. This module includes OpinioNet, Reseval, Iterative ranking, and Homophily Weighted Citation Count (HWCC).

In this executive summary we briefly summarize the features of the use cases described in this deliverable, as well as the architectural philosophy we followed.

LiquidPub Ecosystem. We have understood that the nature of the LP platform is that of supporting applications that implement the LP vision (such as Liquid Journals, Conferences, Books,

Instant Communities, and Liquid Museums) and a knowledge bus that integrates information across (and allows communication among) the different pillars, and that also act as integrated repository of liquid knowledge. In terms of implementation strategy, we progressively understood that the various LP applications that ultimately deliver the LP vision *evolve over time* as we have already seen during the life of the project. One of the main reason for the evolution of the applications is that only once they become adopted we start learning how people use them. We feel this evolution is natural as the concepts are novel. Therefore, each paradigm and supporting tool (including in particular the UI and the design of the interaction with the user) has its own requirements and abstractions. It is important that, at least initially, the abstractions and the conceptual model are tailored for each of them, because this makes it easier to provide a coherent and understandable set of concepts at the appropriate abstraction level. It also makes easier to write code and design the user interaction. However, the architecture also contains a knowledge bus that provides a model for liquid knowledge at a lower level of abstraction and more general in that it needs to cover the concepts of the applications (and of the ones that will come along in the future, to the possible extent). So the challenge in the knowledge bus and its model is to be generic enough to enable integration and extension but also specific enough to facilitate the development of services for the LiquidPub ecosystem.

Liquid Journals. We have identified a model and developed a (prototype) tool for liquid journals, that embodies many of the principles discussed above and specifically i) it supports efficient dissemination of multi-faceted knowledge; ii) it facilitates knowledge search and consumption, that is, the navigation through knowledge (thanks to the linking done by the reader, transforming readers in providers of knowledge); iii) it supports an innovative form of knowledge evaluation, which can be complementary or alternative to peer review, that leverages the community by capturing what each of us does implicitly every day in terms of selecting and sharing papers and using this information as source of evaluation. We have also understood (and modeled) that contextual knowledge about scientific artifacts (relationships among artifacts) can be subjective and we need to allow readers to specify this. We see this as a fundamental shift in the way knowledge is created and consumed. The model has already received great interest by other information providers and publishers/societies.

Liquid Conferences. We have developed a model and implemented a platform for LiquidConferences in complete concordance with the LP philosophy. As a SKO itself, a liquid conference is constituted by a set of contents, authors, panelists, reviewers, readers (registered users), as well as one or more administrators; each of the roles being associated with a set of rights. Each conference is organized as a standard conference on invitation only. Invited speakers are alerted automatically, as they are “created” as authors in the platform and invited to submit their SKO. Then, the SKO will remain open to discussion for a certain amount of time (usually two weeks) and then archived and available for reading only. The temporal dynamics allow for an efficacious interaction among participants, keeping both the lively aspect of the face-to-face events and the standards of written scientific papers. Further, we allow for a modular organization of papers as well as sub-atomic credit attribution to parts of papers. Several liquid conferences, including the Interdisciplinary Workshop on Trust and Reputation took place on Interdisciplines already.

Liquid Books. We have identified a model for publishing books that is evolutionary, collaborative, and multi-facet. The model goes towards LP objectives in that it allows to more easily

build on knowledge of others and to more efficiently release new and updated content. It aims at reducing the barrier that discourage authors from publishing or updating their book and it allows to have books that are very tailored to the needs of the readers. We have identified the main issues we need to address to realize the model, which are related to contracts and to defining processes for credit attribution. The model has already gained the interest of societies and publishers, and we are now doing a pilot with Springer.

Instant Communities. We have identified the detailed list of features, user stories, screenshots and implementation details of instant communities are available at <http://open.instantcommunities.net/>. The application is being piloted in a number of upcoming conferences and is currently used by in a few test universities, with IEEE and with major companies as support for knowledge collection during their internal meetings. 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.

Contents

1 Overall Architecture	1
2 Scientific Resource Space Management System	6
2.1 General Architecture	6
2.2 Data Acquisition	7
2.3 Data Model of SRS	8
2.4 Scientific Knowledge Objects	9
3 Knowledge Spaces	9
3.1 Knowledge Spaces Basics	11
3.2 Knowledge Spaces Platform and Services	13
3.2.1 Architecture	13
3.2.2 Collecting and posting knowledge	14
3.2.3 Intentional Language	15
3.2.4 Other services	16
3.3 Knowledge space Applications	16
4 Evaluation and Reputation Services	17
4.1 OpinioNet	18
4.2 Iterative Ranking	20
4.3 Reseveal	21
4.4 Homophily Weighted Citation Count	24
5 Process and Lifecycle Management Services	25
5.1 Charms	25
5.2 Gelee	28
6 LiquidPub Applications	32
6.1 Liquid Conferences	32
6.1.1 Introduction	32
6.1.2 What is a Liquid Conference?	33
6.1.3 Modularity and multiple authorship	33
6.1.4 Interdisciplines: an implementation of Liquid Conferences. Beta-version	35
6.2 Liquid Books	40
6.2.1 Introduction to Liquid Books in a Nutshell	40
6.2.2 Conceptual Model	43
6.2.3 Related work	44
6.3 Instant Communities	46
6.3.1 Introduction	46
6.3.2 Conceptual model and architecture	47
6.4 Liquid Journals	49
6.4.1 Introduction	49
6.4.2 Scientific Resources for Liquid Journals	51
6.4.3 Liquid Journal Model	55
6.4.4 Creating and Filling a Liquid Journal	56

CONTENTS

6.4.5	Usage model and evaluation metrics	58
6.4.6	Liquid Journal platform architecture	60
6.4.7	Related Work	60
6.4.8	Liquid Journals and LiquidPub	61
6.5	Liquid Museums	62

1 Overall Architecture

The text of this section is reused from the D5.2v2 [15] in order to keep this deliverable self-contained.

The goal of LiquidPub is to capture the lessons learned and opportunities provided by the Web and open source, agile software development to develop concepts, models, metrics, and tools for an efficient (for people), effective (for science), and sustainable (for publishers and the community) way of creating, disseminating, evaluating, and consuming scientific knowledge. This is realized through concepts and models which are supported by an IT solution (the LiquidPub platform) and an ecosystem of services. The LiquidPub conceptual architecture in Figure 1 illustrates main aspects that support LiquidPub vision:

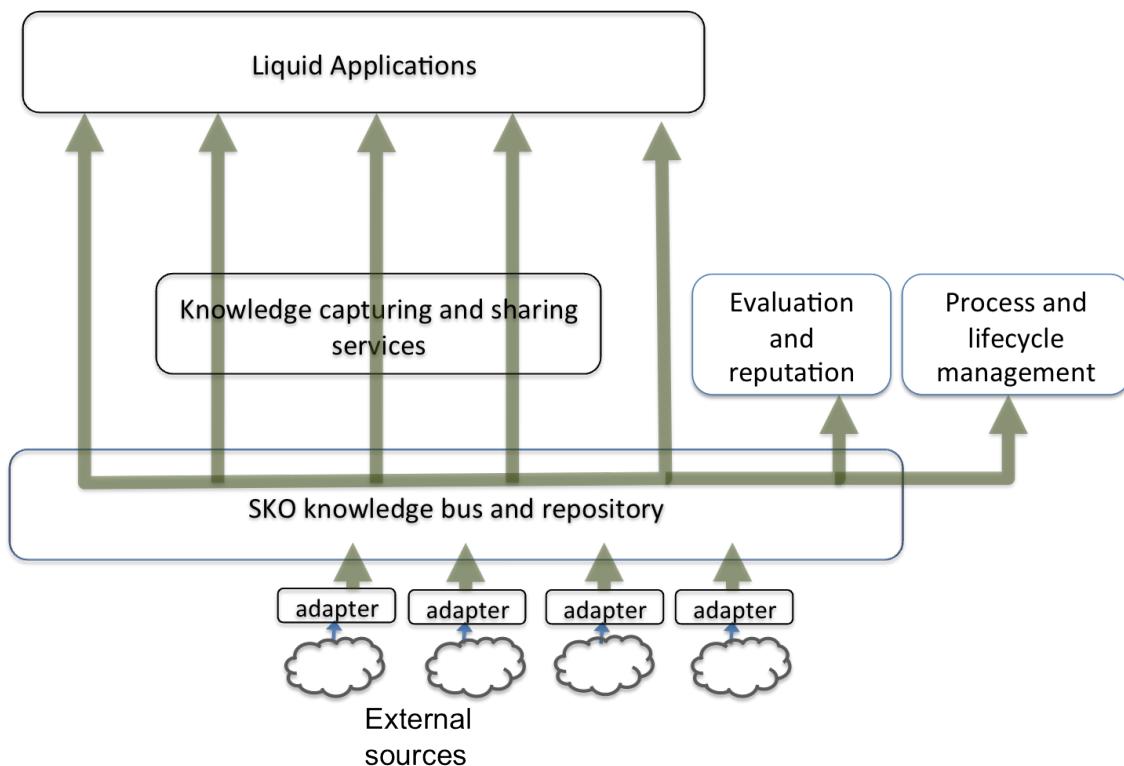


Figure 1: LiquidPub conceptual architecture

- A set of *Liquidpub applications* support different ways of capturing, sharing, and carrying on a *conversation* about knowledge.
- A set of *Knowledge capturing and sharing services* provide access rights and grouping of SKOs
- *SKO knowledge bus and repository* stores and manages SKOs and connects LiquidPub with the world of publishing by means of *adapters* to the external sources.
- An *Evaluation and reputation* module computes metrics on SKOs and researchers

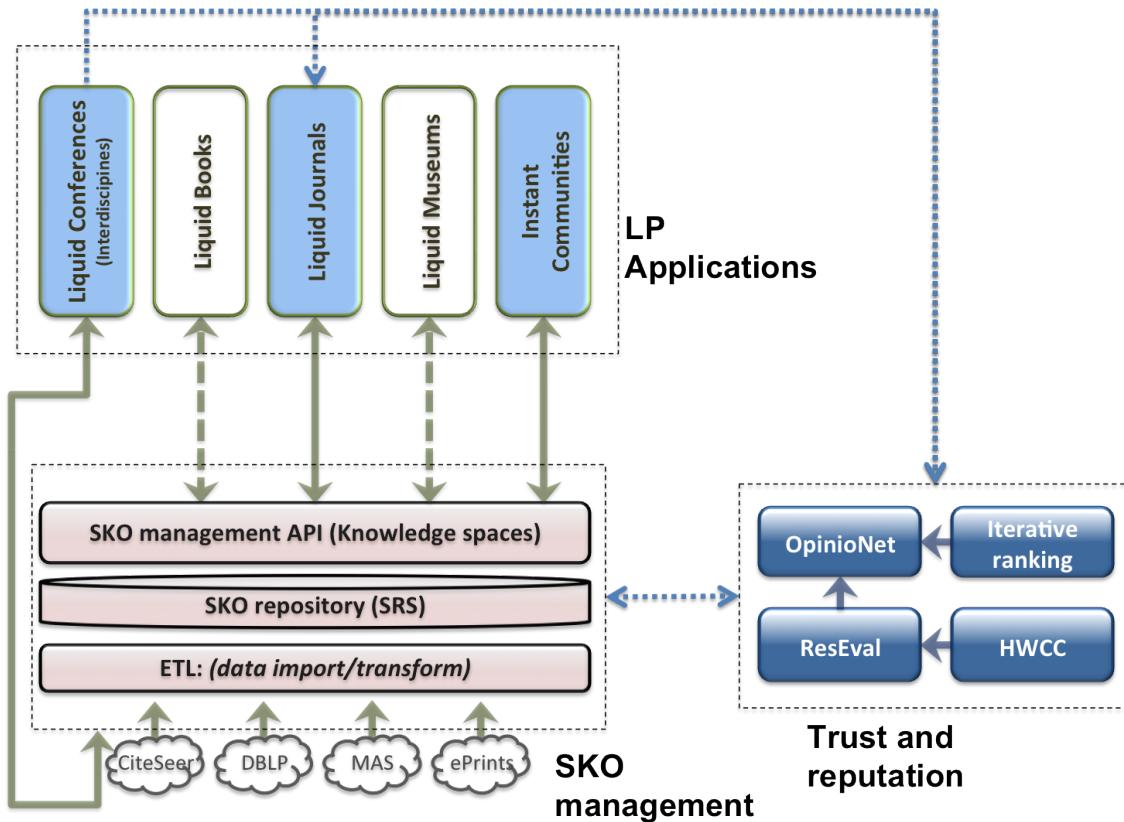


Figure 2: LiquidPub platform. Shaded rectangles represent implemented applications, while white rectangles show applications being implemented. Blue dotted arrows show how the flow of the reputation information, while the solid (or dashed, for non-implemented parts) lines show the flow of the information about the SKO. Finally, blue solid arrows in the trust and reputation module show the flow of the trust information

- *Tools for process and lifecycle management* (Charms and Gelee) support the modeling and definition of the various dissemination processes and lifecycles that can be associated with the liquid artifacts handled by the knowledge bus.

A more detailed architecture of the LiquidPub platform is shown in Figure 2 and is described in the following:

- There are many different contexts in which *LiquidPub applications* can capture and disseminate knowledge, and the applications provide metaphors and interaction patterns for the different contexts in which one can capture and disseminate knowledge. For example, *liquid conferences* facilitate conversations on papers (or parts of a paper), while community-oriented applications like *instant communities* aim at capturing implicit knowledge about scientific contributions and sharing it within a community of interest. Within the LP platform, the applications are represented as the vertical rectangles in Figure 2.
- A Scientific Resource Space Management System (SRS) stores and manages SKOs and allows access to them. It includes

- A SKO repository, that models information collected by the different applications, and also acts as an integrated repository of liquid knowledge.
 - A SKO management API, called Knowledge Spaces, that allows both to access SKOs as well as ways to group them and define access rights.
 - Extract Transform Load (ETL) routines and wrappers that connect LP with existing sources of information and in general with the non-LP world. An example of this consists in getting metadata about scientific publications from external sources such as CiteSeer, DBLP, Microsoft Academic Search (MAS) and ePrints via API or wrappers to the LP platform so that such information can be integrated into the SKO repository or the applications, or can be used by the reputation tools.
- A *trust and reputation* module that processes information from the SKO repository and from the applications to compute metrics that (i) reflect the collective opinions of scientists over scientific contributions, (ii) go beyond metrics with well-known flaws such as citation counts, and (iii) expose metrics that encourage behaviors that are beneficial for science. This module includes OpinioNet, Reseval, Iterative ranking, and Homophily Weighted Citation Count (HWCC).

While advancing with the design and implementation of the applications, we progressively understood that in terms of the implementation strategy, the following aspects have to be considered:

1. The LiquidPub applications and other parts of the LiquidPub architecture that ultimately deliver the LP vision to the user *evolve over time*, as we have already seen during the life of the project. One of the main reason for the evolution of the applications is that only once they become adopted we start learning how people use them. We feel this evolution is natural as the concepts are novel. As we dive deeper into the development and as we have early prototypes available, we better learn and understand requirements as well as how to refine and extend the basic abstractions and underlying conceptual models. We cannot imagine getting the things right from the very beginning. For instance, Instant Communities contains many features that we prototyped in Liquid Journals and envisioned in LiquidBooks. We did not even foresee such tools in the beginning of the project.
2. There is no unique way to capture and share knowledge. We do it differently in different domains. Consequently, there is no single interaction model or metaphor that can fit all needs. However, there are commonalities that we can exploit in the ways we model knowledge, knowledge sharing, and knowledge evaluation-related actions.
3. Researchers in the area of Science 2.0 may develop other dissemination paradigms, or paradigms targeted to specific artifacts (or, we may find over time that many concepts from books, conferences and journals will merge - as we indeed already started to experience with Liquid Museums and Instant Communities). For example, some researchers outside LiquidPub (from UNSW Sydney) developed a *Liquid Benchmark* module, while others are interested in using Liquid Journals as a way to annotate, tag, and link resources in a digital library *they own*, for example a library of *demos* (as in the Share¹ system from TU

¹<http://is.tm.tue.nl/staff/pvgorp/share/>

Eindhoven). These two modules developed by other researchers can already interface with LiquidPub.

These observations carry two important implications. The first implication is that it would be a mistake to make the abstractions of each of the applications tightly coupled. In fact, both from a conceptual perspective and in terms of implementation, in LP we proceeded by developing the separate dissemination paradigms for each application. Each paradigm and supporting tool (including in particular the UI and the design of the interaction with the user) has its own requirements and abstractions. It is important that, at least initially, the abstractions and the conceptual model are tailored for each of them, because this makes it easier to provide a coherent and understandable set of concepts at the appropriate abstraction level. It also makes easier to write code and design the user interaction. For example, Liquid Conferences have the notions of *conference*, of *paper* to be discussed, of *comments*, etc. Liquid Journals have the notion of *scientific resources* that are the items in a *journal*, the notion of journal itself, the notion of *issues*, *editors*, etc. Liquid Books have *books* and *editions* as first class objects. Instant communities have concepts such as *panels*, *panelists*, *presentations*, or *questions*.

Furthermore, as the tools are developed and used, and even as our thought process proceeds and new ideas and opportunities come to mind, these conceptual models and, more in general, the functionality offered by each of the applications evolves. Because of the research nature of the project, binding the models and implementations tightly from the start would reduce the flexibility and make it more difficult to capture the lessons learned during development and early usage.

However, we *do* want to have an integrated end result because we do want each LP application to benefit from what the others can provide. While we believe that a certain degree of autonomy is necessary as it facilitates flexibility and evolution of ideas (and tools), it is also important to connect the applications so that they can share knowledge and put the functionality of one at the service of the other. For example, a paper in a liquid conference can become a scientific resource in a liquid journal or can be shared with a community.

The second implication is that we need to provide the hooks for other researchers to build on top of our results and to be able to easily integrate their idea with LiquidPub ideas and tools. We envision different kinds of extensions, some will be directly extending the applications, but others can build directly on top of the SRS and Knowledge Spaces, as discussed below. This is what we mean by the “ecosystem”: providing a way for other R&D efforts to integrate with LiquidPub (such as Liquid Museums, in development with the Cambridge Museum of Archeology), leveraging what we can offer in terms of liquid knowledge but also providing information that can be collected as SKOs, measured by the LiquidPub reputation metrics, and provided as information to the applications.

These problems (integration and extensibility) are the LiquidPub counterpart of the well-known *enterprise application integration* (EAI) research and development area, where the goal is to integrate a company’s IT systems and services. EAI is needed in an enterprise because systems are developed independently (and it is important that it is so) but then they also need to be integrated, otherwise it is impossible to execute business processes efficiently as these invariably span many IT systems. The response to this was the development of a multi-billion dollar industry around such concepts as enterprise middleware, service bus, and the like. LiquidPub needs the same kind of bus, but for knowledge and for supporting knowledge dissemination. This knowl-

edge bus is represented by the SKO management module, where abstractions related to scientific knowledge are integrated and can be transferred across applications. Having a knowledge bus also implies having a model for liquid knowledge at a lower level of abstraction and more general in that it needs to cover the concepts of the applications (and of the ones that will come along in the future, to the possible extent). So the challenge in the knowledge bus and its model is to be generic enough to enable integration and extension but also specific enough to facilitate the development of services for the LiquidPub ecosystem.

Besides being a bus that allows interactions among applications, the SKO management module includes a repository for those applications that do not maintain one and wish to reuse the central repository, like instant communities and liquid journals. These applications have their own UI and even the supporting API that has application-specific concepts and metaphors. This API then invokes the (application-independent) knowledge spaces API thereby mapping specific concepts into SKOs and SKO management operations. Examples of the architecture of the LiquidPub application are given in Section 6.

The trust and reputation module has the same underlying principles: some aspects can be made generic and as such insist on the the SKO repository and SRS. Other parts are inherently application-specific and integrate with the applications (also because they have grown and were elaborated with the application and depending on the information that the application provides). For example, the OpinioNet reputation module was developed for the general case of reputation handling and then separately integrated with Liquid Conferences and Liquid Journals applications. OpinioNet takes information from the Iterative Ranking algorithms and from the applications and provides reputation information back to the applications directly, while Reseval exploits the integration of SKO metadata at the knowledge bus level. This means that reputation in LP entails mapping reputation-relevant information not only from the applications, but also from the external sources such as DBLP and Microsoft Academic Search into the shared model and then operating on this to derive reputation metrics for people and scientific artifacts.

What this all means in terms of architecture and in terms of LP development process (development of both concepts and software) is the following:

- The applications initially had their own abstraction and conceptual model. These were also the initial requirements for a knowledge bus/shared model.
- Over time, some of the applications converge towards the common model as represented in the SRS, at varying degrees of integration. For example liquid journals and instant communities still have an own UI and a thin API layer, but now insist on top of the same Knowledge Space API and SRS modules for SKO management. Liquid conferences are also integrated via an adapter though they also maintain their own repository.
- Over time the shared model evolves to accommodate new requirements at the appropriate level of abstraction. In other words, in some cases new concepts in the application may simply correspond to new mappings, while in other cases may result in new abstractions to be pushed down to the level of the shared model to SRS or Knowledge Spaces level.

This strong emphasis on *applications* (which we see not only as use cases, but rather as key elements of what of LiquidPub is exposed to users) was not part of the initial description of work.

We also observe that the evolution of our thinking in terms of architecture means that what we have is not really a main platform with plugins to be added (the initial thinking), but rather full-fledged software applications to be integrated, many of which even have their own UI to interact with the end users.

The deliverable has the following structure. In Section 2 we describe the Scientific Resource Space Management System, while Section 3 describes the Knowledge Spaces tool. Section 4 and Section 5 describe the reputation and process and lifecycle management tools in the LiquidPub architecture. Finally, in Section 6 we describe five examples of the LiquidPub applications of different degree of maturity.

2 Scientific Resource Space Management System

2.1 General Architecture

The central component of SRS is the Metadata Warehouse (Figure 3), whose implementation largely follows the traditional ETL (Extract Transform Load) process. The Adapter Layer encapsulates the differences in the data sources and their data models, and helps to cope with the heterogeneity of scientific metadata. Each adapter is responsible for getting metadata according to the protocols and APIs provided by the source and transforming it into the model of Scientific Resource Space². This task is performed in a few steps. First, scientific metadata is gathered from a source by the corresponding adapter and stored into a so called source dump - set of preliminary tables dedicated to the source. This step is executed periodically, and each time we try to recognize and store only the elements appeared or updated since last execution. The metadata is then loaded into the staging area where it is joined with metadata from other sources. At this stage, metadata elements from each source are preliminary merged based on the identifiers provided by source. This ensures that we introduce no duplicates at the source level, while there may still be duplicated instances of the same metadata elements coming from different sources. The following cleaning phase mainly focuses on solving this problem. During the cleaning phase the staging area is analyzed to discover (entity matching) and merge entities duplicated across different sources. The algorithms of such entity matching may vary from simple and intuitive ones, such as comparing titles of the scientific papers, to potentially sophisticated ones like analyzing the co-authorship graphs of scientists. The current approach to matching papers compares their titles, venues and publication years. The authors of the merged papers are also merged after their names are matched with special string comparison rules taking into account name abbreviations. After being cleaned, the metadata is finally loaded from the staging area into the target database, where it is made available for the applications. Similarly to the previous phases, the loading is performed by computing and making the changes with respect to the current state of the target database.

The applications built on top of SRS focus on different aspects of the scientific resource metadata. In order to provide useful functionality with the reasonable performance, they require efficient access to their own presentation of the scientific resource space. For instance, Reseval - a tool for evaluating scientific contributions, authors and institutions - operates various research metrics including publication and citation-based ones. The number of citations and self-citations

²Note that providing uniform access to scientific resources on the Web, SRS now replaces the ResMan tool described in D5.1v2.

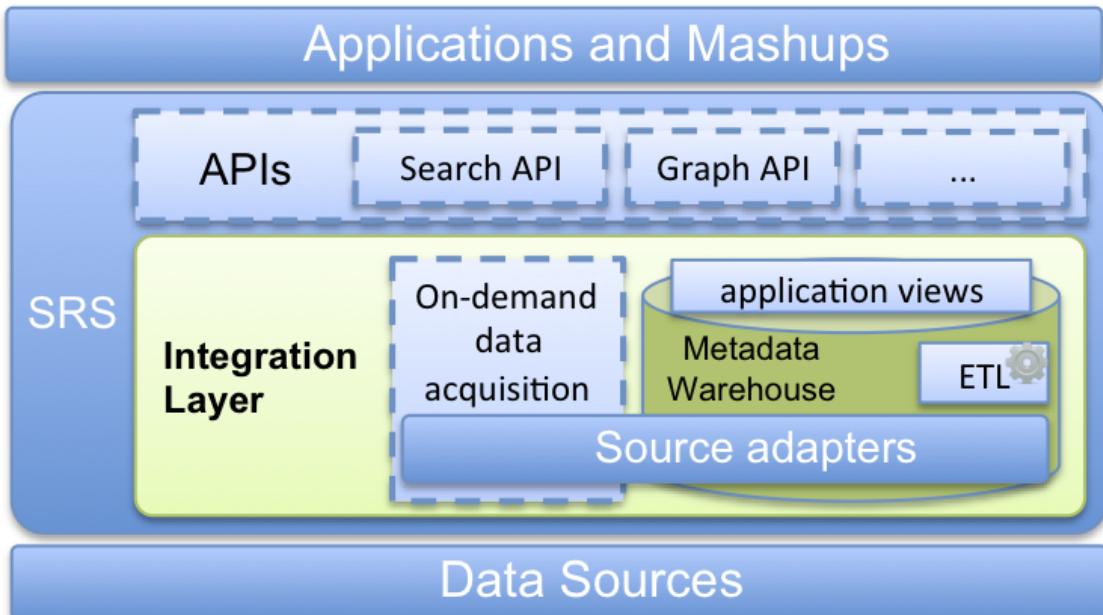


Figure 3: The architecture of SRS (SKO repository)

of a paper or an author are the primary units of data for Reseval, accessed very frequently and used to construct more complex metrics. For efficient access, these numbers can not be calculated dynamically and have to be precomputed. SRS addresses this problem by creating the application-specific views which contain all the data needed by the application in a suitable format, and are updated at the final stage of the ETL process. These application-specific views, as well as general functionality of SRS can later be exposed to applications via the set of APIs, such as search API or API for navigating the graph of scientific resources. These APIs are currently under development.

In order to enable source-dependent requests, SRS propagates the information about the sources of metadata elements through all the stages of the process to the target database and the application-specific views. At any time for any metadata element it is possible to learn which source or sources it originates from. This allows applications like Reseval to compute metrics such as h-index with respect to any source or combination of them.

Apart from warehousing of metadata, SRS is also looking to support on-demand data acquisition from sources. This approach potentially allows our services to employ sources that expose only search interface, and also to provide personalization in cases when sources rely on user profiles. The opportunities of on-demand data acquisition are currently being under investigation.

2.2 Data Acquisition

SRS acquires its data from the sources via set of adapters, which wrap the sources to encapsulate the differences in their data models. Depending on the APIs provided by sources, adapter implementations may require reading an xml data file, or calling REST services or sometimes fetching and parsing web pages. Currently the three main sources of data for SRS are DBLP (Digital Bibli-

ography & Library Project), Microsoft Academic Search (MAS) and Interdisciplines. DBLP gives out their complete dataset as a plain XML file that is periodically updated. The corresponding adapter reads the file with the help of SAX parser and extracts bibliographic records. In particular, the data about publications, their authors and venues (conferences or journals) is obtained and stored into the database of SRS. Microsoft Academic Search is approached by fetching its web pages of the publications, and extracting the data from them via HTML parser. Apart from data about publications, authors and venues, this source also gives us the references of the publications. Adapter for Interdisciplines connects directly to their database via jdbc and retrieves data about publications, authors and venues.

The specific functionality of the adapter for Microsoft Academic Search includes looping through ids of the publications in MAS, parsing the content of its web pages and storing the extracted data in dedicated tables. The common functionality reused also by other adapters includes establishing HTTP connections with the server and getting the content of web pages, while ensuring non-intrusive communication with the source.

2.3 Data Model of SRS

We see scientific contributions as 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 today's 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.

Formally, we define the space of scientific resources as $\sum = \langle SR, E, L, A \rangle$ where

- SR is a set of resources $r = \langle id, uri, ct, cf \rangle$ 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.
- 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 $l = \langle es, et, lt, u, un \rangle$ 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 users or agencies that endorse it, if any, $un \in E$.
- A denotes a set of annotations $a = \langle e, at, v \rangle$ 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”).

2.4 Scientific Knowledge Objects

As explained on D1.2v3, and evidenced when comparing Figure 1 and Figure 2, the concepts of the SKO conceptual model and its role as a Knowledge Bus and repository are realised through SRS.

More specifically, SRS can be called a Knowledge Bus because it aggregates several heterogeneous external sources while also translating them into a single model. It also stores this information into its internal database constituting an updating knowledge repository of the scientific resources that are connected to it (e.g. DBLP, Interdisciplines). Furthermore, it also organizes this information into views according to the needs of the high-level ’pillars’ (e.g. Liquid Journals, Liquid Conferences) of the project.

The model of SRS is also a concrete implementation of the SKO model presented in D1.2v3, tailored for the representation and unification of scientific resources. In more detail, the Set of Resources (SR) explained in subsection 2.3 is used to represent resources while Links (L) is used to represent the relations that exist between them (one of such relations can be the ’part of’ relation, thus representing the aggregation of resources). Annotations (A) are also added to capture custom comments metadata to be added to the resources. Note that all the previous concepts are analogous to the ones presented in D1.2v3, with the following particularities:

- The distinction between wholes (i.e. full documents and scientific resources, SKOs) and parts (i.e. components of the document, SKOnodes) of resources was not explicitly implemented in SRS as, for the moment, none of the LiquidPub applications required complex services with this distinction (although this may change for eventual future applications).
- Through its Entities (E) attribute, SRS also represents the actors involved in the different processes related to the scientific resources. This fairly straight-forward name-based specification, was also suggested as necessary in the SKO theory with a more in-depth treatment being considered for future work.

3 Knowledge Spaces

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 virally. 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 (and this is why we select the web engineering conference as outlet). 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. Mendeley, 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.

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 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 algorithms for helping scientists find the knowledge they need, and of UI and interaction metaphors that facilitate its usage.

3.1 Knowledge Spaces Basics

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 discuss and formalize the concept of Kspaces.

A Knowledge Space is defined as $KS = \{R, Q, M, Tr, C, S\}$, i.e., a collection of *SRS* content (Figure 4), with the following characteristics:

- The content is defined intensionally³ (in terms of the properties the content should have) or extensionally⁴ (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 this section.
- 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.

³http://en.wikipedia.org/wiki/Intensional_definition

⁴http://en.wikipedia.org/wiki/Extensional_definition

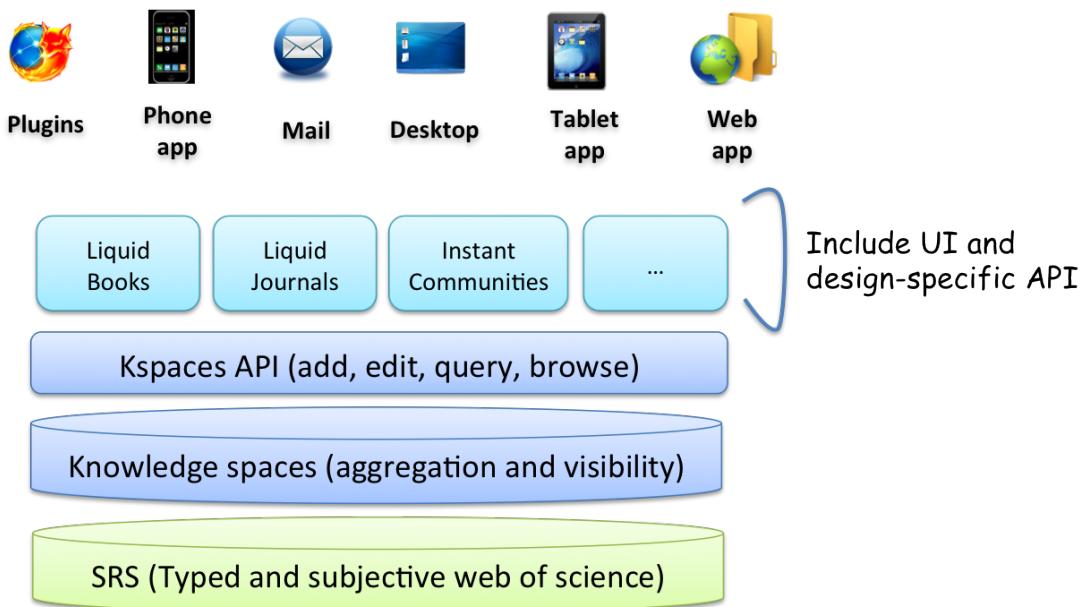


Figure 4: Kspaces and KS designs are ways to create, share and consume resources

- $Tr = \{transparent|opaque\}$ 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 what is 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 specific 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. Note however that the definition of resource, annotations and relations is managed by the underlying SRS model.
- 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. Note that the KS is aware only of the current state, thus allowing the lifecycle model to be decoupled and managed

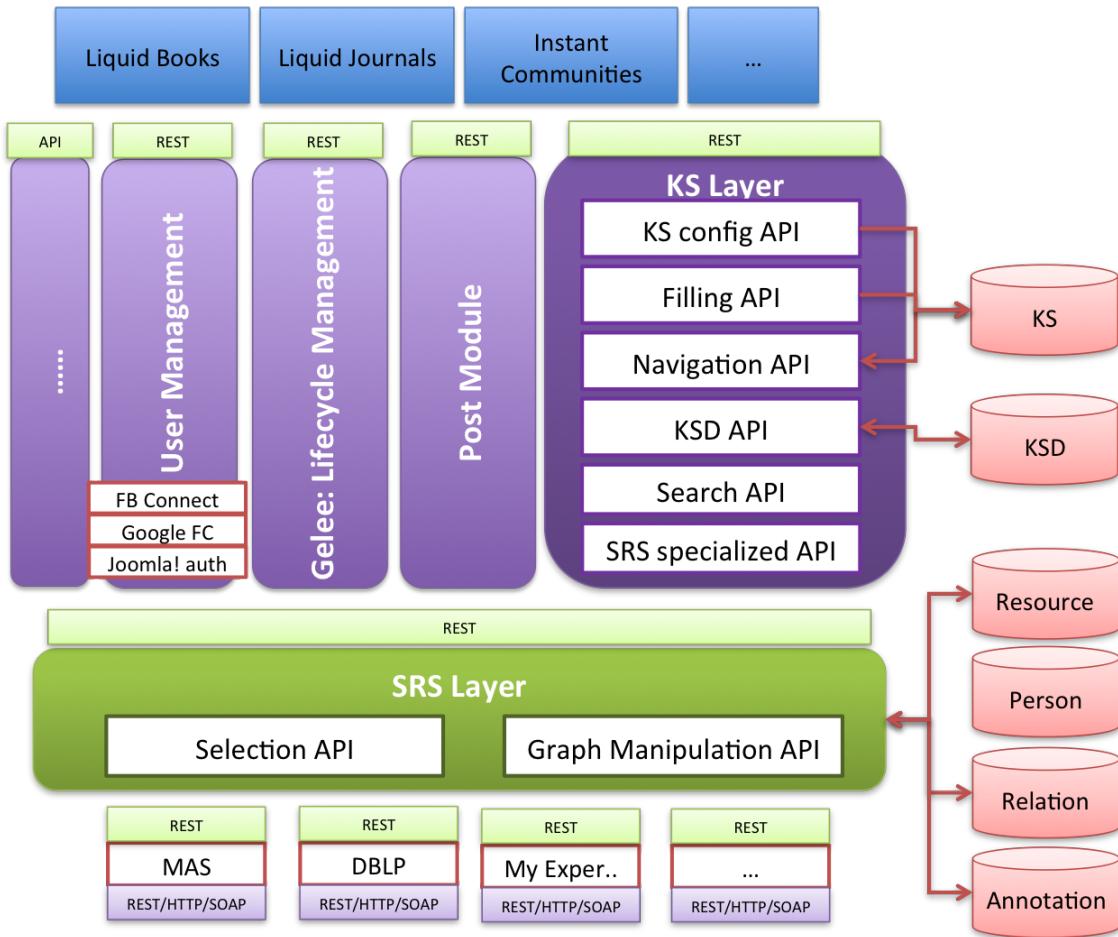


Figure 5: Architecture of the KS platform

externally.

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.

3.2 Knowledge Spaces 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. Details are available at <http://services.kspaces.net>.

3.2.1 Architecture

The high level architecture of the platform is described in Figure 5.

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 [3].

Above the SRS, the *Knowledge Space Layer* implements the KS primitives and exposes them as web services. It also implements the interface 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 [5] 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.

3.2.2 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)

3.2.3 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 6). 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. An example of navigation is that a 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 some specific sources. For example, some users might want to get articles only from Springer (a certified scientific resources source) others would prefer only papers appearing in Google Scholar (a more comprehensive but noisy source). This is implemented in the example UI in 6 as a filter on the left side of the workspace.

In addition 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 [2]). 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 6 and are available for testing at <http://journal.kspace.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.

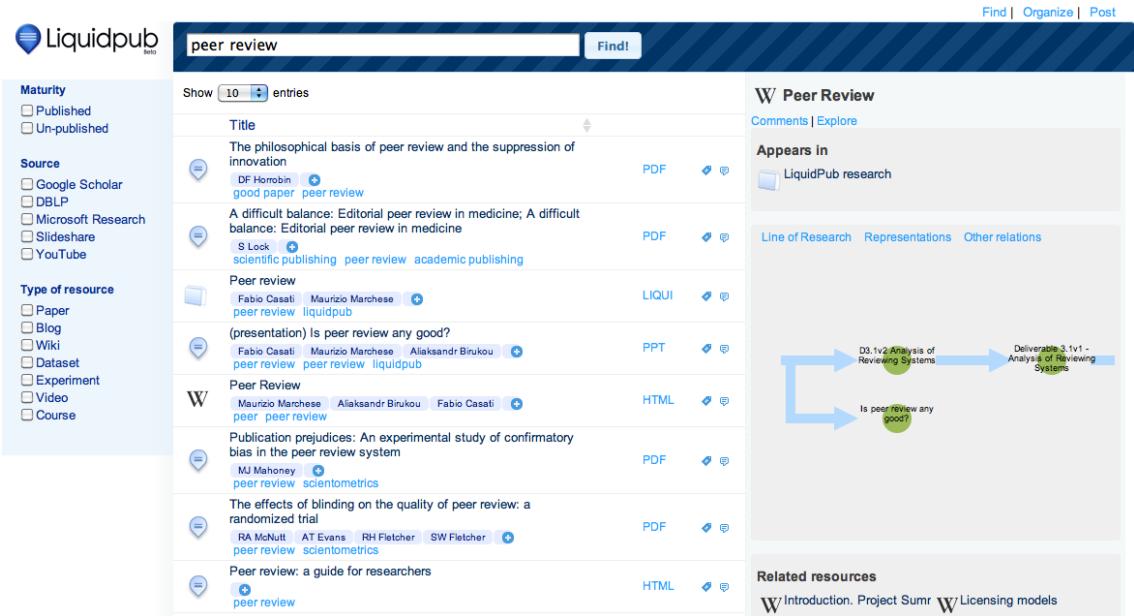


Figure 6: Example of Interface using KS intensional language

3.2.4 Other services

Besides post and intensional selection, Kspaces offer other services not further detailed here, such as search, export in specific formats, serialization of the content (e.g., to create a single pdf out of a set of resources), lifecycle management, reputation in the social scientific web and the like. The implementation of these services is at different levels of maturity. Details can be found at <http://services.kspaces.net>.

3.3 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 (described in Section 6.3) 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 intentionally 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.

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.

4 Evaluation and Reputation Services

This section⁵ presents models and architecture of the reputation modules developed in LiquidPub: OpinioNet, the iterative ranking module (ItRank), Research Impact Evaluation (Reseval), and the homophily weighted citation count (HWCC). OpinioNet computes the reputation of both researchers and research work (or SKOs) based on the innovative ‘opinion propagation’ algorithm developed in WP4 and getting input from the LiquidPub applications and repositories. ItRank helps rank reviewers based on their performance in past reviews. The results of ItRank may be used by OpinioNet for calculating the researcher’s final reputation measure. Reseval, on the other hand, is a web application that allows people to view a variety of reputation metrics and compare them. It also computes “traditional” reputation metrics such as citation counts and h -index. OpinioNet uses Reseval’s metrics as initial reputation measures, before taking into consideration additional sources of reputation, such as reviews. Finally, the homophily weighted citation count (HWCC) helps provide metrics that distinguish between intra-community and inter-community citations to give more weight to the latter in order to promote diversity. The integration of these tools is described in details in D5.2v2 [15].

4.1 OpinioNet

The OpinioNet reputation module⁶ provides a set of algorithms for computing the reputation of two of the resources defined in the LiquidPub Platform: contribution (or a SKO) and person (or a researcher). In both cases, reputation is computed depending on which information sources contribute to this calculation. The table in Figure 7 shows the types of reputation and the different information sources considered by OpinioNet. The reputation of additional roles (e.g. collaborators, invited speakers, etc.) as well as additional knowledge related entities (e.g. libraries, institutions, etc.) may also be investigated in a similar manner; although we leave these issues for future work.

The reputation of a scientific contribution is computed in terms of the opinions the contribution has received. The reputation of researchers, however, depends on the role that researchers play. We currently distinguish between two roles that can be played by researchers inside the LP world: author and reviewer. The reputation of an author is based on the reputation of its contributions. However, the reputation of a reviewer may also consider an analysis of his/her reviews or his/her social network. ItRank (see the next section) would be helpful here since it helps rank reviewers based on their past reviews.

OpinioNet is wrapped into a RESTful web service and integrated in Reseval as a new pluggable metric (see Figure 8).

The Representational State Transfer (REST) Interface for computing the reputation metric is the following:

⁵The text of this section is partially reused from [14] to keep this deliverable self-contained

⁶<http://project.liquidpub.org/reputation/>

reputation of		information source
contribution		contribution's opinions
person	author	author's contributions
	reviewer	person's reviews
		person's social network

Figure 7: Reputation module: types of reputation

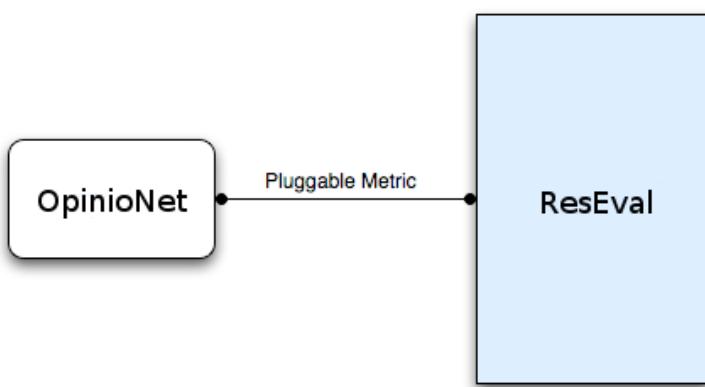


Figure 8: OpinioNet as a pluggable metric in Reseval

http://eru.iiia.csic.es:9090/ReputationService/resources/compute_reputation/Object/ReputationEntity/ReputationMetric/MetricOption

where the input parameters represent:

- **Object:** Id or name of the object to be evaluated; it could be the name of a contribution or a person.
 - **ReputationEntity:** The name of the entity to be evaluated. For the time being only four options are possible: liquid journal, contribution, author, and reviewer (although the reputation of authors and reviewers are equivalent in the current implementation).
 - **ReputationMetric:** In this case we want to compute the reputation based on a given metric. For the time being the options are the following: contopinions (to make use of the opinions about contributions), authorcont (to make use of a researcher's contributions), revreviews (to make use of a researcher's reviews), or revnetwork (to make use of social relations between researchers).
 - **MetricOption:** an option of how reputation will be calculated (for instance, based on citations, opinions, etc.).

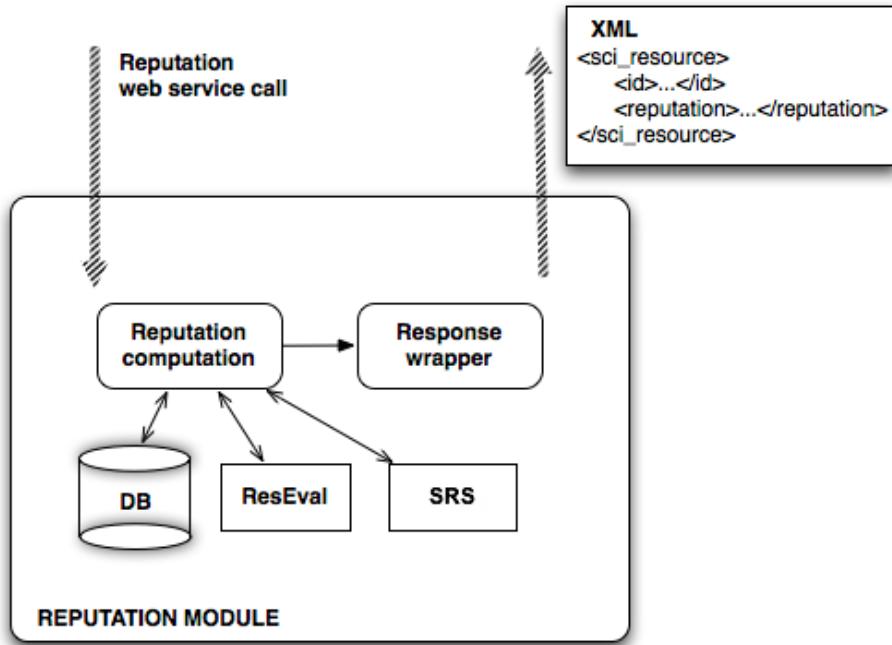


Figure 9: The architecture of OpinioNet

Following, an example of a computation request for a contribution and its result is given. This link computes the reputation of a contribution with id = 10: http://eru.iiia.csic.es:9090/ReputationService/resources/compute_reputation/10/contribution/contopinions/none.

The result of the reputation service invocation will be an XML with a number representing the reputation of the object evaluated. The data to compute the reputation is retrieved from different sources such as Reseval and SRS. A complete scheme of the reputation module can be seen in Figure 9.

Implementation A detailed description of the algorithm is presented in Chapter 4 of Deliverable 4.1 [12].

The API specification of the OpinioNet is available in Google Docs at: <http://docs.google.com/Doc?docid=0AVLiEhF9ZeIFZGhoM3FyeD1fMGpqbXd0NGho&hl=en&invite=CJH7-dYB>

The code of the OpinioNet reputation module can be found at http://project.liquidpub.org/reputation/code/propagation_demo.zip.

4.2 Iterative Ranking

The role of the Iterative Ranking module (ItRank) is to assign weights to individuals evaluating scientific content (reviewers) and provide rankings of the evaluated content. We use a particular class of algorithms (so-called co-determination algorithms) which use given evaluations to obtain a unique aggregate rating of the objects. These ranking algorithms search for a consensus in quality of every object and penalize the users who differ from this consensus in their evaluations. By iterating this consensus-penalization procedure over time, ratings from malicious or unskilled users can be weeded out, providing both a better estimation of object quality and an enhanced overall reputation-based ranking of objects. Note that by objects we mean objects of the rating. In practice, this may be an actual object (a book or a scientific paper), a person (an eBay auctioneer), other entity (an online seller), or an SKO (scientific knowledge object) which is particularly important in the context of the LiquidPub project. Implemented algorithms were evaluated on artificial datasets [10], real datasets [11] and generalized to multidimensional ratings [13] (which is particularly important in the context of science where submissions are often judged by their clarity, novelty, overall contribution, etc.).

User weights obtained by ItRank may be used directly as reviewer reputation by OpinioNet [12] and, conversely, additional information provided by OpinioNet may serve to improve user weights. ItRank takes only a set of triplets user-object-rating⁷ outputs a vector with user weights and object quality values. A detailed description of the algorithms is presented in deliverables D3.2, D3.3 and D4.2 [10, 11, 13].

The source code is available at: <http://project.liquidpub.org/reputation/code/ItRank.zip>

4.3 Reseval

The Research Impact Evaluation (Reseval) module ⁸ is in charge of managing and computing the metrics of scientific entities, such as authors and contributions. The architecture of Reseval is defined in three well-differentiated layers in Figure 10. Those layers are the typical ones that are defined in a three-tier architectural pattern, namely the user interface, the business logic (core) and the data layer. From the client point of view we have implemented two types of interfaces, to grant both web access through “classic” web pages, and automated access using the REST interface, which is the most commonly adopted solution for web services. The choice of developing multiple interfaces is motivated by the need to have, eventually, the possibility to embed the application in more complex platforms and applications. Since during the design phase the flexibility requirements have been taken into account, the system is ready for changes or extensions required for its inclusion in other systems. The caching system on the data level is clearly motivated by performance reasons, since we are dealing with a web application that often queries external services, such as Google Scholar and, therefore, loses much time due to network’s overhead.

Interface layer: this layer provides the main functionalities of the tool through several user interfaces. The metric management interface offers all the functionalities needed to create new metrics or update existing ones. The metric computation interface allows one to compute existing

⁷In the case of multidimensional ratings, multiplets user-object-multidimensional_rating are expected instead.

⁸<http://reseval.org>

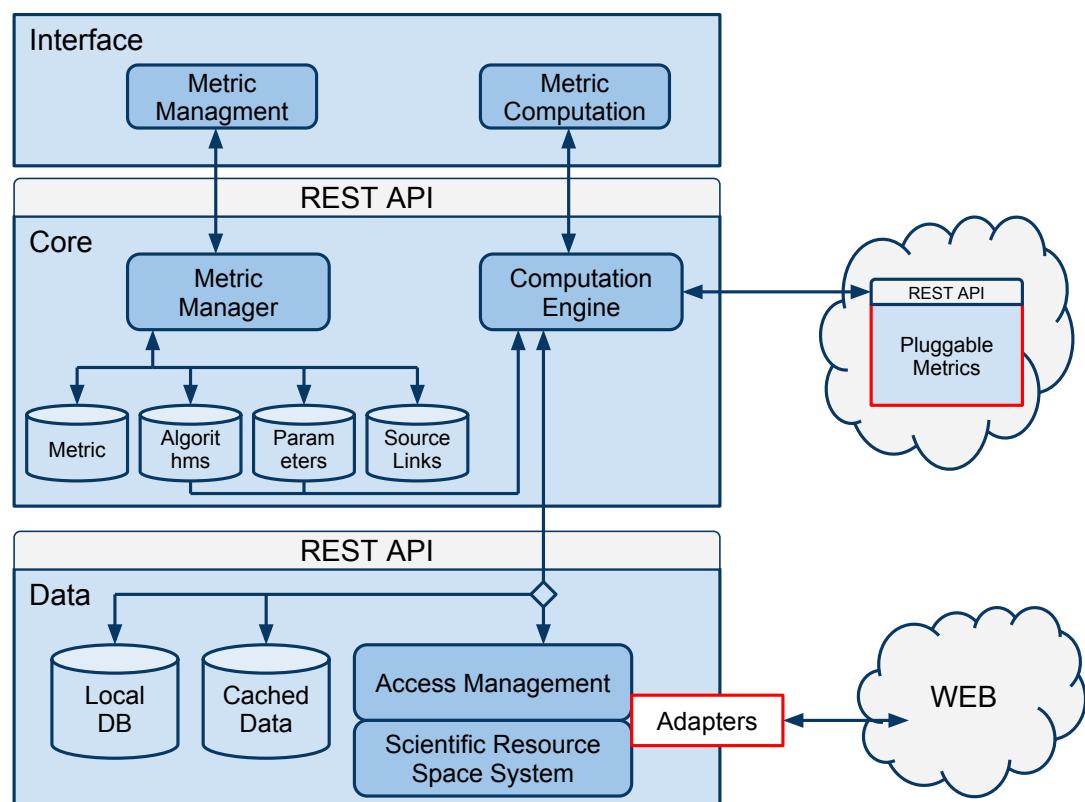


Figure 10: The architecture of Reseval

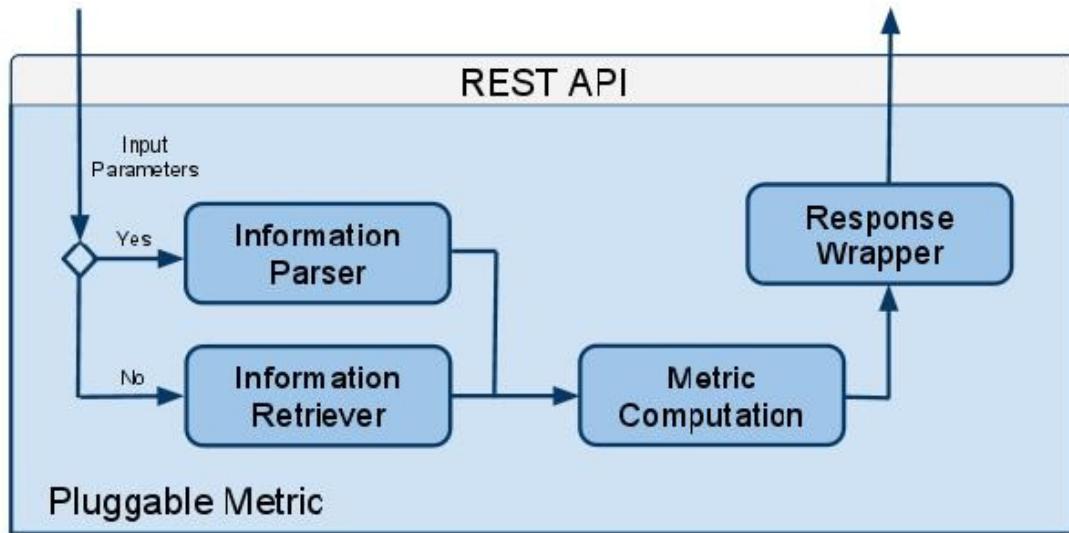


Figure 11: Pluggable metric architecture

metrics defined using the available options and information source (e.g., Google Scholar).

Core layer: this layer stores and manages all the definitions and logic of the metrics. Metric manager is the core component of the tool, it that allows one to create, update and delete metric definitions. Computation engine computes the metric using the specified algorithm and its parameters that indicate metric definition and information source. The modules that can be identified in the metric architecture, and that can be seen in Figure 11, are the following:

Information parser: is the module that parses the information received from the application and prepares it for the metric computation.

Information retriever: is the module that gets the information from the corresponding source and prepares it for the metric computation.

Metric computation: is the module that allocates and executes the algorithm that computes the metric.

Response wrapper: is the module that prepares the response message that will be returned to the application with the result of the metric.

The example of a call and its parameters to the REST interface for computing a metric is following.

BaseURL/resources/compute_metric/object/objectType/metric/metricoption/source/sourceoption/reload/publist.

Following is the details of these input parameters:

- **Object:** name of the object to be evaluated.
- **ObjectType:** type of the entity that is evaluated (author or contribution).
- **Metric:** name of the metric that will be computed.

- **MetricOption:** this represents the algorithm that will be used to compute the metric.
- **Source:** is the link to the repository from where the input parameters have to be retrieved.
- **SourceOption:** is the filter to be used when the information is retrieved from the specified source.
- **Reload:** is the flag that indicates that recent information is requested from the source instead of cached one.
- **Publist:** is the parameter telling if user wants to load publication list or not, “yes” to load and “no” to not.

An example of a valid call to the REST interface is `http://demo.liquidpub.org/reseval_v3/resources/all_metrics/maurizio%20marchese/author/Google%20Scholar&as_subj=eng/no/no`. This link computes the h-index of Maurizio Marchese using Google Scholar as a data source.

Data layer: this layer is used to get the data needed in order to compute a specific metric. There are different sources where the tool can get the data: internal sources, such as a local database, or external sources found in the web. For these external sources an adapter must be implemented that is able to retrieve the information for each source. Furthermore, the adapter must be registered in the infrastructure modules in order to be available for use. The architecture includes the cache mechanism to store the information retrieved from underlying sources. The caching system is clearly motivated by performance reasons, since we are dealing with a web application that often queries external servers and, therefore, loses much time due to network's overhead.

Implementation The source code of the Reseval tool is available at `https://dev.liquidpub.org/svn/liquidpub/ResEval/`. Please use lp-guest and lp-password credentials to access it. The API specification of the Reseval RESTful service is available as a Google Doc at `http://docs.google.com/Doc?docid=0AUULLw3NbестCZGM4em54YndfMTZkZ2Q5cmdmYg&hl=en`. Guidelines on how to use the API with examples are available at `https://dev.liquidpub.org/svn/liquidpub/papers/deliverables/Accessing%20ResEval%20Restful%20API.docx`

4.4 Homophily Weighted Citation Count

The concept of homophily describes relationships that are based on some measures of similarity or closeness. We defined a family of bibliometric indicators, namely the homophily weighted citation count that takes similarity to be inversely proportional to the distance that separates researchers in the co-authorship network and papers in the citation network. The rationale behind this is to provide a new family of metrics with an indicator that favors publications having an impact in broader communities. Basic concepts of these metrics were presented at the Interdisciplinary workshop on Trust and Reputation hosted online. A copy of this article, with all the concepts explained, is available at `http://www.interdisciplines.org/paper.php?paperID=111`.

The computation of these metrics is exposed through an external Web Service API hosted at the University of Tartu. This API is wrapped within Reseval in order to expose it through the common API of Reseval.

Homophily metrics Web Service is implemented using a REST architecture style. It provides three basic operations to:

1. Launch the computation of the Homophily Weighted Citation Count (HWCC) of an Author/Venue.
2. Poll the result of the computation of the HWCC associated to a given Author/Venue.
3. Delete the HWCC of a given Author/Venue using the resource created by operation (1)

According to the REST principles, the operations are requested using HTTP verbs, the endpoint and input parameters are encoded in URLs. The results are in turn encoded in HTTP header metadata (e.g. Status Code, Location) and response body. In this document, interactions are described with UML sequence diagrams. The tool is freely available at <http://curl.haxx.se/>. The Web Service provides end-points for handling queries about HWCC of authors, i.e. <http://ats.cs.ut.ee:8080/HWMetrics/resources/hwccAuthors>, and another one for venues, i.e. <http://ats.cs.ut.ee:8080/HWMetrics/resources/hwccVenue>. However, the set of interactions remains the same.

Homophily metrics are also integrated to the API of Reseval, that obtains and wraps the result of the external service that provides these metrics.

5 Process and Lifecycle Management Services

This section⁹ reports on Charter Management System (Charms) and Gelee System. These are tools developed in LiquidPub and dealing with the management of research and dissemination processes. In particular, Charms deals with the specification and the automatic execution of the processes concerned with the creation, dissemination, and evaluation of research work. Charms deals specifically with formal and repeatable processes, such as a process of a conference (submitting articles, reviewing, submitting camera ready) or of a journal submission, and manages the progression of such processes. The novelty of Charms is that it automatically adapts the execution stage's user interface according to the modification of the process specification, if any. Gelee is instead a tool for lightweight lifecycle management. It allows users to define the lifecycle of any artifact identifiable by a URI (e.g., SKOs such as deliverables, papers), to monitor its progress, and to automatically execute actions on resources upon the resource entering specific lifecycle states. A typical example application is to define and monitor a quality plan for the artifacts. Gelee does not include an engine (the state is advanced manually) and the lifecycle can be defined and evolved “as you go”, at runtime. It merely provides a guideline for execution that can be followed or even disregarded. As such, it is intended to be used in more informal and liquid processes, such as the collaborative writing and reviewing of a deliverable.

⁹The text of this section is partially reused from [9] to keep this deliverable self-contained

5.1 Charms

Charter Management System (Charms)¹⁰ provides users with straightforward means for the specification and the automatic execution of the processes concerned with the creation, dissemination, and evaluation of research work. In other words, the aim is a user friendly system for managing research work.

The specification is done by means of charters. A Liquid Publication charter outlines the conditions under which a certain process that is concerned with the creation, dissemination and evaluation of research work is organized. A Liquid Publication charter (from now on referred to as charter) is defined by a set of rules. For instance, rules could address who is allowed to create, modify, assess scientific knowledge. Rules may also specify deadlines for actions, the accepted order of actions, and so on. We refer to these rules as the specification of the charter. Examples of charters are creating and running conferences, managing projects, writing papers collaboratively, compiling books, and so on.

In addition to the specification of charters, the management system should also provide the automatic creation and update of a web-based user interface for the related researchers (or personnel). Existing examples of this are current conference management systems, such as ConfMaster and EasyChair. However, while existing conference management systems are hardcoded and the interface requires a manual update every time a change occurs, the Liquid Publication management system proposes an automated generation and update to the web-based user interface, eliminating the need for manual modifications.

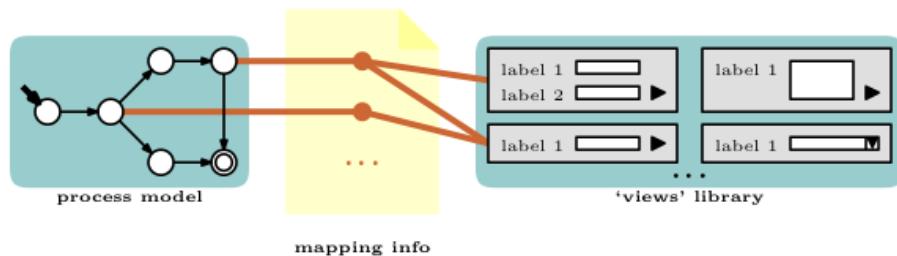


Figure 12: Elements of Charms

To achieve the objectives above, the user needs to specify the charter and enrich it with information about user interfaces as shown in Figure 12. Ideally, we aim at providing a straight forward user interface that would allow the user to produce the specification through a simple drag-and-drop. Hence, we say the charter can be specified as a transition graph and an additional mapping file is needed to map the states of the transition graph to a set of 'views'. This is because each peer playing a specific role in the charter model should see a different set of views, based on its current state in the model. Naturally, a library of 'views' should also be provided to allow the user to choose from.

To implement the above, we make the following choices:

- charters are implemented as Electronic Institutions (*EI*) [16]. *Islander* editor, one of the

¹⁰<http://project.liquidpub.org/charms/>

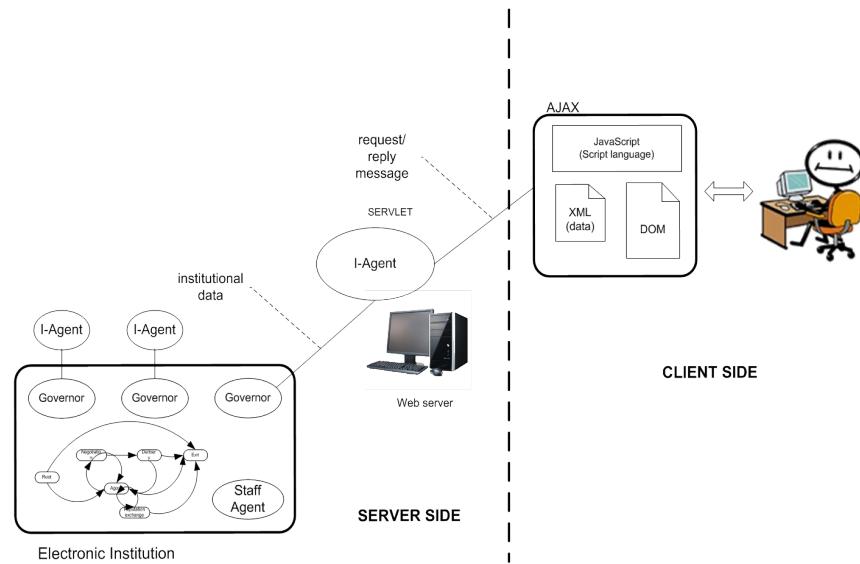


Figure 13: Architecture of Charms

tools of Electronic Institutions Development Environment (*EIDE*) [1], already provides an easy to use user interface for the specification of process models through a simple drag-and-drop. Once charter model is specified, it can be executed using *AMELI*, also provided by *EIDE*.

- the library of views will essentially be a library of widgets
- the file mapping between states of the charter and a set of views is specified in XML

Figure 13 shows the main elements of Charms architecture:

Electronic Institution. Is the center of the system which provides the infrastructure for defining and executing charters.

Web server. This is the main link between a human user and the EI.

Staff agents. Institutional agents in charge different aspects related to the well-functioning of charters.

I-Agents. Agents that represent human users (charter personnel) in the EI. They also act as web servers (servlet) and connect to EI by interacting with the **governors** through institutional events.

Mapping info. Defines a map among charter states, charter user roles and views. That information is used by I-agents to update properly the web-based user interface.

Client application. We have adopted an approach based on two web technologies: Java Server Pages (JSP) and Asynchronous JavaScript And XML (AJAX). JSP technology provides a simplified, fast way to create dynamic web content by merging HTML with JAVA.

In contrast, AJAX is considered a web design technique that provides a high level of interactivity, avoiding the undesirable reloading of the web pages after each human user action. Hence, the client application only needs a relatively recent web browser and therefore it is not necessary to install special software.

The I-Agent, acting as a servlet, receives information requests from the web browser of the user. I-Agent replies to the web browser according to the information containing in the mapping file and the relevant changes that have been produced in the EI.

Implementation The code of the Charms can be found at <http://project.liquidpub.org/charms/Charms-0.2.zip>.

5.2 Gelee

Gelee¹¹ is a tool for modeling and managing lifecycle of web artifacts. The main principle of Gelee is to keep the lifecycle model simple, with only a few elements and constructs. This simplicity extends to the underlying architecture thus enabling a lightweight infrastructure. We first define the overall architecture, and then discuss two among the most important aspects: action-resource interaction and lifecycle widgets. Although they are “embedded” in the architecture section we draw the reader’s attention to it as they correspond to key decisions in terms of keeping the model simple, and in terms of usability and reusability.

The main concepts behind this work are drawn in [4].

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 14 depicts the high-level architecture, composed, essentially, of three layers: the data tier, the kernel and the user interface.

The functionalities offered by the Gelee Core are provided by four sub-components:

- **Design-time manager:** It manages the lifecycle modeling and templating. Lifecycle models and templates are versioned and can be shared among groups of users.
- **Runtime manager:** The runtime manager manages the progression of lifecycle instances, performs change management and derives the action executions to the basic services layer.
- **Monitor:** The monitor subcomponent allows getting the trace and status of lifecycle executions.
- **Configuration manager:** It manages the user base, access control of lifecycles, and the configurations of the lifecycle management component.

¹¹<http://project.liquidpub.org/gelee/>

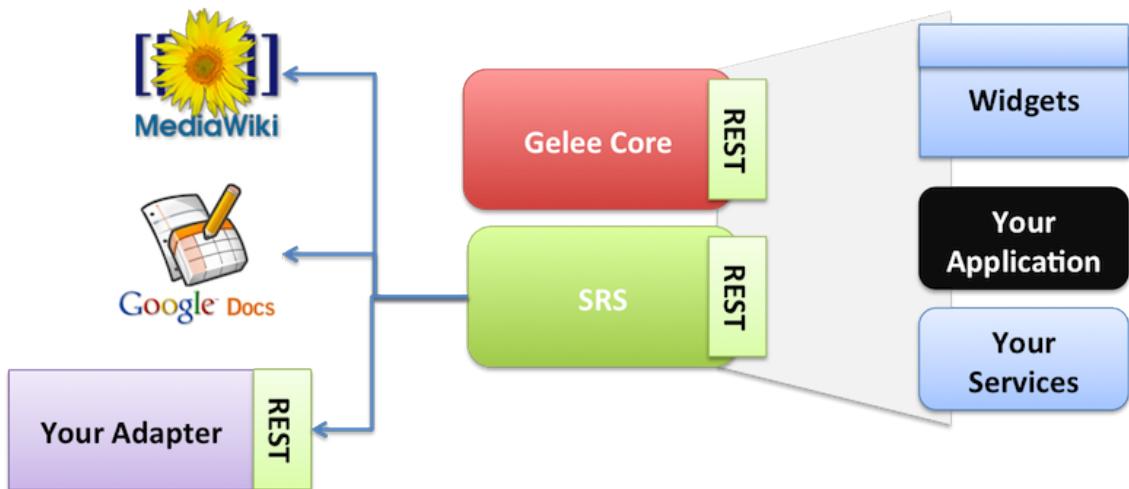


Figure 14: The architecture of Gelee

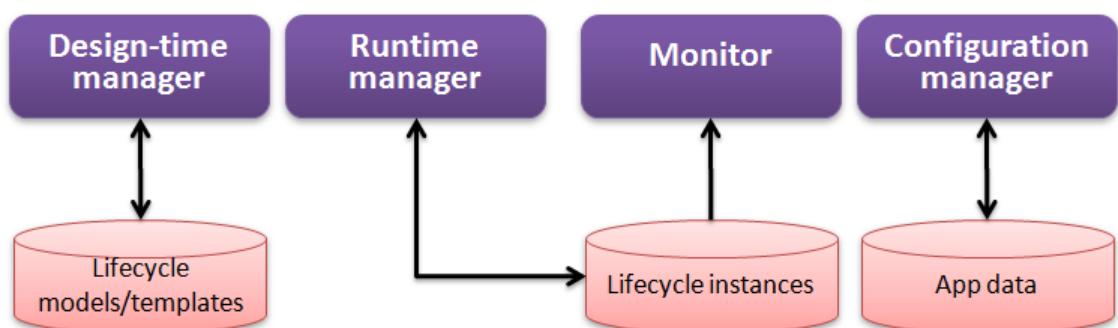


Figure 15: The architecture of Gelee Core module

The following services provided by the Gelee Core are exposed through a REST interface:

- Creating, updating, reading, deleting, searching lifecycle models and lifecycle model templates (CRUD + search). Note that by updating lifecycle models and templates we provide versioning.
- Creating, updating, reading, deleting, searching lifecycle instances (CRUD + search).
- Migrating lifecycle instances. This is performed by updating the current state of the lifecycle instance with a new state in the target lifecycle model. A state in this context refers to a set of properties including a reference to a phase.
- Performing a transition. This is performed by updating the current state of the lifecycle instance with a new state in the same lifecycle model.
- Getting a lifecycle instance trace. This service allows getting the execution log.
- Creating, updating, reading, deleting, searching users. It allows managing users in the lifecycle management workspace.
- Creating, updating, reading, deleting, searching groups of users (CRUD + search). It allows users to organize themselves in groups to share models and templates.

The interfacing between Gelee and a specific resource occurs through plug-ins or adapters offered by the Scientific Resource Space management (SRS) described in Section 2. 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 in the SRS, 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 to handle all actions in a standardized fashion. Gelee also provides support for lifecycle modeling and execution through the use of the resource and actions definition and managing access rights.

Lifecycle Widgets. Gelee offers UI-level integration through resource type-specific widgets and web interfaces. Web UI are provided for the lifecycle designers Figure 16 and for the monitors. Both offer an AJAX-style interface, easy and immediate to use.

Execution of lifecycles is instead different, as ideally it is integrated with the resource, for example, it is shown in the same web page, or as a browser plug-in. For this aspect we use widgets. Widgets are components ready to be integrated with web applications or even desktops. Through widgets, users see the lifecycle and the resource they manage side by side, as shown in Figure 17.

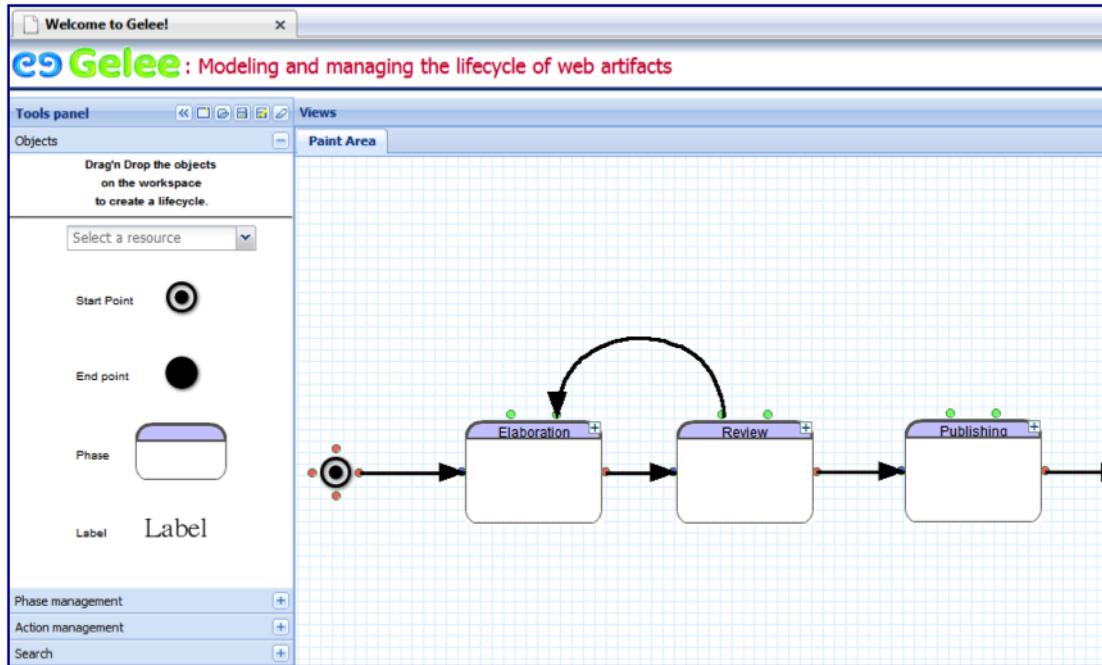


Figure 16: Modeling interface

The integration between lifecycle management and the resource, as shown in Figure 17, is done by combining execution features provided by the lifecycle manager interface, and resource-specific information provided by the resource manager. The latter offers the interface by which we can render any resource in a transparent way.

Widgets are also subject to visibility attributes. Attributes like access rules are automatically auto-discovered from the lifecycle definition. Thus, a user interacting with a widget could be requested for authentication or not based on the visibility attributes, and also, different users could have different views of the same lifecycle (i.e., managers, resource owners, and stakeholders in general).

Users may have more specific needs beyond the services we provide through our widgets. Therefore, we allow them extending from the API to develop their own widgets or web components (e.g. a Facebook application). Moreover, because of the added value of composing the services from different source, we prepared our widgets to put in pipes (e.g. Yahoo Pipes). For example, users could feed our widgets with Google Docs feeds listing documents, and use that list to reflect the lifecycle of those documents.

Implementation The code of the Gelee is available at <https://code.launchpad.net/gelee>. The information for Gelee developers is available at <http://project.liquidpub.org/gelee/developers.html>. In particular, the API specification is available at http://bazaar.launchpad.net/~cdparra/gelee/trunk/view/head:/docs/gelee_reference.pdf.

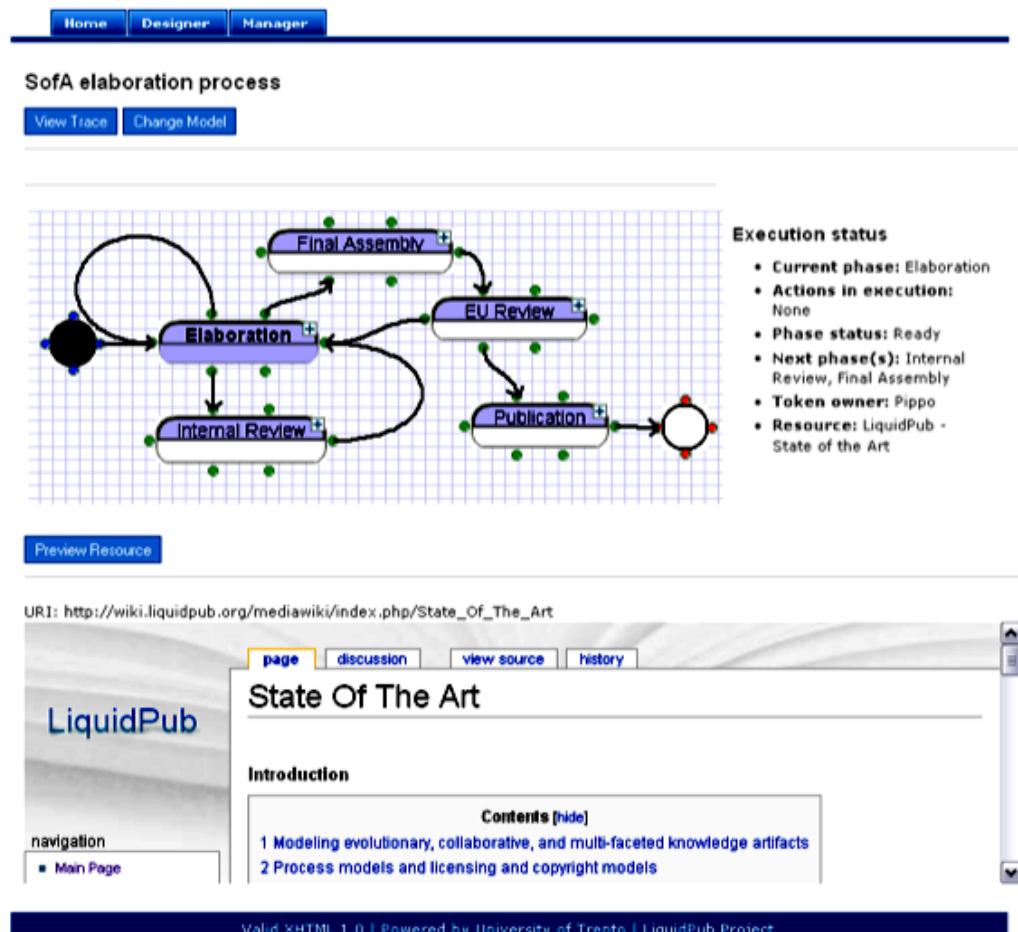


Figure 17: Integrated Resource Lifecycle Management widget

6 LiquidPub Applications

We now discuss the five LiquidPub applications: Liquid Conferences, LiquidBooks, Instant Communities, Liquid Journals and Liquid Museums.

6.1 Liquid Conferences

6.1.1 Introduction

Research¹² papers are ways of keeping active a conversation among researchers, whose roles and accesses can be different according to the position they hold towards the SKO that is communicated. The dynamic of standard academic exchanges in the “solid” world of classic publications is that of a conversation in slow motion: too long messages communicated at a too slow rate of transmission. The success of the email among the community of researchers shows how needed was a technology to quicken the exchanges. Nevertheless, this acceleration of learned conversations among researchers faces many institutional, social and technical obstacles due to the weight of copyright policies, research practices, like peer review, and old technologies of publication. One of the goals for the second year of the LiquidPub project was to rethink the dynamics of learned conversations by reconceptualizing roles, accesses, licensing models and providing adapted tools able to encompass the different needs of the accelerated conversation. We provide a description of roles and licensing models in [8], and this section presents an example of dynamic tool for research exchanges: Interdisciplines, implementing main aspects of the Liquid Conferences use case. In the following we will outline the structure of a liquid conference, which aims of solving research and communications problems in science related to standard models of publication. We show how single issues have been tackled and partially solved by the beta-version of the Interdisciplines platform we put online.

6.1.2 What is a Liquid Conference?

A liquid conference (or a virtual conference) is constituted by:

- A set of **papers** (structured in a “modular” form that will be detailed later in the presentation)
- A set of **authors** (defined by a profile and an institution)
- A set of **discussants**, (defined by a profile and an institution)
- A set of **moderators** (who can edit the papers, edit the profiles of authors, discussants, and reviewers)
- A set of **reviewers** (profiles can be anonymized, but known by the moderators)

¹²Note that an alternative version of this section also appears in Section 3.1 of LiquidPub Deliverable D2.1v2. We consider this to be an example of Liquid Publication in action, with different deliverable authors making collaborative use of the same shared SKO resources.

- A set of **readers (registered users)** (by registering by name and location, they can leave tags and evaluations on papers)
- One or more **administrators** (who can open/close conferences, export them, give/change permissions to all the roles)
- A set of **organizing institutions**.

Each conference is organized as a standard conference on invitation only. Invited speakers are alerted automatically, as they are “created” as authors in the Interdisciplines platform and invited to submit their SKO. Then, the SKO will remain open to discussion for a certain amount of time (usually two weeks) and then archived and available for reading only. The temporal dynamics allows for an extremely efficacious interaction among participants, keeping both the lively aspect of the face-to-face events and the standards of written scientific papers.

6.1.3 Modularity and multiple authorship

One of the main themes of discussion among the LiquidPub team revolved around the modular structure of the SKO and the possibility to assign a more granular authorship to subpart of the SKO according to different levels of collaboration (see D1.1 and [7]). An initiator of a SKO, that is, the person who has the idea of creating a new starting point of a scientific conversation, may decide to assign specific areas of the SKO to collaborators for co-writing the paper. We implemented this need into the new Interdisciplines platform the CNRS team has developed in partnership with the start-up ColDev (Collective Developments). The platform implements most of the ideas behind the Liquid Conferences use case, and the technical details of the platform are explained in Section 6.1.4. The new platform allows for collaborative writing and multiple authorships of modular papers. We will outline the structure of the modular paper, will show some screenshots of the demo, and indicate the URLs of the online platform.

The modular paper. We have conceived a template for modular SKOs in which each section is pre-defined and autonomous. Pre-defined sections are:

- Title
- Keywords
- Abstract
- Introduction
- State of the Art
- Central Claim
- Results
- Methods
- Discussion/Conclusions

The screenshot shows the 'My Account' section of the Interdisciplines platform. At the top, there's a navigation bar with links to Home, Bibliography, Links, Archives, About us, My account (which is highlighted in orange), Register, and Log out. Below that, it says 'Welcome Aliaksandr Birukou'. The main area is titled 'My Account' and contains several input fields and buttons:

- Conference:** A dropdown menu with the placeholder 'Select a conference for this paper first.'
- Title:** An input field.
- Keywords:** An input field with the placeholder 'Separate each keyword using a comma >'.
- Abstract Title:** An input field with a link 'How to create a footnote?'.
- Abstract:** A rich text editor window.
- Introduction Title:** An input field.
- Introduction:** A rich text editor window.
- Manage Related Media:** Buttons for 'Add an image', 'Add a Dataset', and 'Add a Video' (with a note '(Youtube video only)').
- Media thumbnails:** A section with notes about related media and instructions for uploading and inserting media.
- Manage Creative Commons licensing:** A dropdown menu with options 'Attribution by', 'Attribution Share Alike', and 'Attribution No Derivatives'.
- Save Paper:** A yellow folder icon.
- Save & Preview:** A magnifying glass icon.

Figure 18: Structure of the modular paper in Interdisciplines

- Acknowledgments
- Bibliography

Each of these different units can be edited autonomously and can have a specific title. A possible improvement in liquidity that is not implemented yet would be the option to add as many “modules” as needed for each different SKOs, thus leaving even more freedom to the authors. Another improvement we envisage is to have a more structured template for each module, or a series of templates (CSS sheets) that the authors can choose in order to fit their data at best. The structure of the modular paper is visible at <http://interdisciplines.org/main.php?actionType=writePaper>¹³ or in Figure 18.

Multiple authorship. Another innovation of the Interdisciplines platform is the option of managing multiple authorships for each module of the paper. The initiator of the paper (the first author) can add for each module of his/her paper a number of authors, if they are already in the website database, or suggest to moderators/administrators to invite authors. The order of authors’ names is established in the end by the initiator of the paper. His/her name is the default first name, but he/she can decide to change the order by just dragging names one below the other. See Figure 19 or <http://interdisciplines.org/main.php?actionType=coWritePaper>.

Managing licenses. For each new paper, authors can decide the type of licensing model they want to endorse, if it is compatible with the overall licensing model of the website. The list of licenses appears in a menu and authors can easily select the appropriate one. Once the conference

¹³For this and following links to Interdisciplines you must register at Interdisciplines and login

Interdisciplines

Home | Bibliography | Links | Archives | About us | **My account** | Register | Log out

Welcome Aliaksandr Birukou

My Account

Profile Write a Paper Co-Write a Paper Manage Bibliography Summary

Conference
Select a conference for this paper first.

Title

Keywords
< Separate each keyword using a comma >

Abstract Title How to create a footnote?

Abstract

Introduction Title

Introduction

State of the Art Title

State of the Art

Central Claim Title

Manage Authors
This paper was initiated by : Aliaksandr Birukou
Casati Roberto
Brito Ismel

Drag and drop author names to manage their order of appearance.
Invite Author by sending an email to a moderator
(only for Authors that dont already appear in the Assign Authors menu)

Manage Related Media
Add an image Add a Dataset Add a Video (Youtube video only)

Media thumbnails
Please note that Related Media can fit in the Central Claim only.

- Upload your media so that their thumbnails appear below.
- To insert media, place your cursor in the Central Claim and click on a thumbnail.
- Important:** If you finally delete a media file after inserting it, please manually remove its link from the Central Claim.

Manage Creative Commons licensing
Learn more about the CC licensing models
Attribution by Attribution Share Alike Attribution No Derivatives

Hold down the "control" key to select several Licensing models

Save Paper **Save & Preview**

Figure 19: Managing authors and licenses on Interdisciplines

Interdisciplines

Home | Bibliography | Links | Archives | About us | **My account** | Register | Log out

Welcome Aliaksandr Birukou

My Account

Profile | Write a Paper | Co-Write a Paper | **Manage Bibliography** | Summary

First name: <input type="text"/>	Each entry you add appears in the list below.
Middle Name: <input type="text"/>	• Click on the title of a paper to open its bibliography. Click on it again to close it.
Initial(s): <input type="text"/>	• You can delete or edit any entry.
Last name*: <input type="text"/>	• If you click "Edit", the data you wish to edit will appear on the left.
Year: <input type="text"/>	• Once you're done, hit "Save entry".
Title*: <input type="text"/>	
Journal (if article) <input type="text"/>	
Volume (if article) <input type="text"/>	
Issue (if article) <input type="text"/>	
Publisher (if book) <input type="text"/>	
City (if book) <input type="text"/>	
Pages <input type="text"/>	
URL <input type="text"/>	
ISBN <input type="text"/>	
DOI <input type="text"/>	
Assign entry to a paper <input type="text"/> Le Monde est Grand Le Monde est Grand Paper with Cafes Second paper Title1	SAVE ENTRY

© Interdisciplines, 2010  Institut | Nicod   

open sourced and designed by [Collective Developments](#)

Figure 20: Managing bibliographies on Interdisciplines

is closed and the destiny of the papers change, the authors still keep the privilege of updating the license of the paper. You can see the box for managing licenses in Figure 19.

Bibliography. Authors can enter their bibliographical references into the EndNote compatible template available on the website. They can assign the reference just to one paper or to others. They can also add an external link to an online bibliographical tool, such as Mendeley, to point to their online bibliographies. See <http://interdisciplines.org/main.php?actionType=manageBibliography> or Figure 20.

Reviews. Assigned reviewers can write an anonymous review through a template that has been condensed by looking at the practices of the main journals and main evaluation tools for research. See <http://interdisciplines.org/main.php?actionType=readReview> or Figure 21.

Interdisciplines

Home | Bibliography | Links | Archives | About us | **My account** | Register | Log out

Welcome Aliaksandr Birukou

My Account

Profile Write a Paper Co-Write a Paper Manage Bibliography Summary

Write a Review

Please note that all reviews are anonymous communicated to the author(s)

Content
Please, write a brief summary of the main thesis of the article.

Focus
Has the central topic/focus of the paper been clearly articulated?
 Excellent Good Acceptable Serious Problems

Originality
Is the article sufficiently novel and interesting to warrant publication? Does it add to the canon of knowledge? Does the article adhere to the website's standards? Is the research question an important one?
 Excellent Good Acceptable Serious Problems

Arrangement/Logic
Are sections placed in logical, effective order? Are all sections relevant to the paper as a whole?
 Excellent Good Acceptable Serious Problems

Methodology
Was the methodology appropriate for the conclusions drawn? Was the analysis correctly done? Were the results of analysis interpreted correctly?
 Excellent Good Acceptable Serious Problems

Data settings
Were the tables, figures and, if any, videos, appropriate and adequate?
 Excellent Good Acceptable Serious Problems

Writing
Is the writing coherent and clear? Is the language adequate? Has the manuscript been proofread, that is, is it relatively free from misprints and grammar errors?
 Excellent Good Acceptable Serious Problems

Overall Rating
 Excellent Good Acceptable Serious Problems

Evaluate your own expertise in the field of the paper:
 Excellent Good Acceptable Serious Problems

Notes for the editors

Figure 21: Submitting review on Interdisciplines

6.1.4 Interdisciplines: an implementation of Liquid Conferences. Beta-version

In this section we briefly present Interdisciplines, an implementation of the Liquid Conferences use case. Deliverable D5.2v2 [15] and the presentation at <http://www.slideshare.net/ColDev/interdisciplines-20> provide more information.

Conferences, authors, papers, discussions and reviews. Each conference on Interdisciplines gathers a selection of papers written or co-written by registered authors. An active conference is one in which all registered users can discuss, comment, evaluate and review each paper. After the termination of a conference - whose duration is determined by Administrator(s) and Moderator(s) - all discussion and commenting features will be disabled. Nonetheless, the full conference will remain fully accessible to the public as an archive (conference abstract, papers, bibliographies, author biographies, discussions and comments). Finally, papers and discussions can be exported in PDF format at any point in time.

Interdisciplines manages five user types with graduated levels of permissions and access: Administrator, Moderator, Author, Panelist, Registered Reader. All contributors to conferences and to co-written papers are approved by either the Administrator(s) or the Moderator(s): prior to the release of any paper, an author has to be invited and granted permission to participate in a conference. Finally Interdisciplines comprises an evaluation template for papers to be reviewed by all user types. Each review is revised by the Administrator before being transmitted to the reviewed author(s). Reviews are not public and can only be viewed by the reviewer, the reviewed author(s) and the Administrator(s).

Administrative interface and the main page. To visualize the Interdisciplines platform, you can open the administrator's page at <http://interdisciplines.org/admin.php>.

You can also open the authors', readers' or participants' page at <http://interdisciplines.org/index.php> or in Figure 22.

6.2 Liquid Books

The text of this subsection is reused from the LB demonstrator document [20] in order to keep this deliverable self-contained.

6.2.1 Introduction to Liquid Books in a Nutshell

When teaching, professors and students need a book which is tailored for the class, which is up to date, which includes (or points to web-based) exercises, which has a companion web site with additional readings, etc. And, at the basis, a textbook that is correct and clear. Today, this rarely happens: books are written by a small set of authors, require a lot of coordination actions among them and because updating books often requires a big effort by both authors and publishers, then usually new versions contain just minimal changes or more chapters at the end. From the perspective of professors, it is very hard to find a book tailored for their class, meaning that they have to resort to a collection of books, with the related problems (cost, inconsistent description,

[Home](#) | [Bibliography](#) | [Links](#) | [Archives](#) | [About us](#) | [My account](#) | [Register](#) | [Log in](#)

Interdisciplines

A website for interdisciplinary research in philosophy, cognitive science and social science. It allows for the organization of interdisciplinary conferences.

Current Conferences

Governing the Future
Opening date : May 30 2010 00:00 UTC
Closing date : Jul 14 2010 00:00 UTC

This conference brings together different scholars from different disciplines who share a common interest in the study of the future and will discuss the role of predictions in different areas such as the environment, finance and demographics. A specific attention will be paid to the causes and historical reasons for the development of these predictive tools and the social demands these modern oracles respond to. The different papers will underline the role knowledge plays into the elaboration of predictions. Predictive knowledge includes social sciences theories and models or techniques such as computer science or statistics.

The contributors will provide answers to specific questions such as:

- Has the role of forecasting changed over time?
- What is the role of future technologies?
- What is the impact of forecasting?

Recent Comments
by [Birukou](#), Apr 22 2010 13:06 UTC

wow!
[more](#)



© Katrin Vierkant

© Interdisciplines, 2010  Institut | Nicod    

open sourced and designed by [Collective Developments](#)

Figure 22: Main page of Interdisciplines

difficulty in exposing a coherent set of topics in a coherent way, difficulties in linking topics and classes and exercises, etc.).

In the LiquidPub project we have explored and proposed the concept of Liquid Books (LB) as collaborative, evolutionary, possibly open-source and multifaceted versions of the traditional books (either printed or digital). Liquid books (LB) are collaborative, evolutionary, *multi-facet* versions of the traditional books. In a nutshell, a liquid book is characterized by the following entities (in italics):

- *content* added by *contributors*
- A content is published to the public via *editions*, prepared by *editors* (editors are a subset of contributors).
- a *contract* defines who can add or consume content, who can create editions, what is the frequency of editions. A contract defines also the licensing/visibility policies for content among contributors.
- A contract also includes the specifications of a number of *processes*, e.g. for modifying the pool of authors (including new authors or removing previous ones), defining credit attribution mechanisms etc.
- a *publisher* publishes and distributes the editions of the book, typically holds the copyright, and possibly provides additional services, including IT services supporting LBs.

A traditional book is one instance of the model above. In a traditional book (think of authored books, not of edited books for now) we typically have one or a very small number of editions, typically not in parallel. All contributors (the authors) are listed as authors of the book, and get royalties based on a pre-agreed contract. Variants of the book are typically subject to new contracts. In particular, an author cannot just take the content and publish its own variant without contractual agreements with the other authors and with the publisher. Also, in a typical publishing process, there are rigorous quality checks. The “time to market” is overall quite high, both because it takes a long time for authors to prepare the manuscript and because of the publishing process after the initial manuscript has been delivered, that also takes several months. Overall this resembles very much a waterfall software development process, with all its benefits and limitations.

The opposite extreme of a liquid book resembles more agile and open source software development, with some twists due to the nature of book publishing and copyright management. So, at the “liquid extreme” we have a liquid book where contributors add content to the LB, possibly in a continuous fashion, and where contributors can freely decide at any point to prepare new editions by collecting and editing content put by other contributors to the same LB. With such flexibility, each edition could be easily tailored for a set of specific readers. At the proper phase, the associated processes (as defined in the contract) starts. These processes can range from very simple rules (e.g., editors always take 20% of the royalties and the rest is shared equally with all contributors) to more complex procedures (for example, a voting mechanisms that approves royalty distributions proposed by the editor). The key here is to have these processes simple and well defined, at the beginning, otherwise the complexity is excessive and very likely this will not scale.

The main goal behind the LB concept is to enable authors to easily share and reuse their content, giving them at the same time the guarantee that their content will be used appropriately

and so producing versions of the book tailored for the needs of the prospective readers (students, consultants, etc.) LB can be open source, partially open, up to the case of traditional closed book. In our model we aim at covering all the possibilities, leaving to authors the final choice of how to distribute their content.

A primary - but not the only - pilot scenario we have focused on is that of educational textbooks, but all the concepts described for this scenario can be also applied to different ones, like major reference work (e.g. Encyclopedia) where larger pools of authors are involved and where the need for update is very strong.

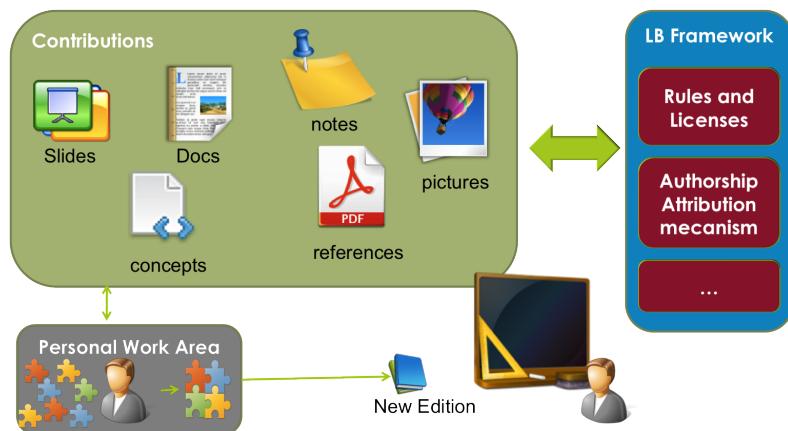


Figure 23: Liquid Book Textbook Scenario

The main concept behind a LB textbook is to let professors compose and create a customized (and as liquid, evolving) textbook for their class by putting together modular (but "refinable") content from other colleagues. A LB textbook is an evolving and multi-author collection of materials on a given topic that unlike traditional books/e-books, is composed by different kinds of artifacts (documents, slides, exercises, code, experiments etc.), is collaboratively written by several, possibly (but not necessarily) cooperating authors, and has various author/publisher defined copyright models (see Figure 23). The idea is to have a few professors that share their teaching materials (slides, sections on given topics, exercises), as they want to produce a book for their class, and then each one of them or some of them produce a different version of the book, as they may have a different vision on how to organize the content. Then we envision that each professor at the start of the class will "create" a version of the book that is tailored for his/her class by reusing the shared material, either composing it when possible, or taking it and modifying it when needed - and when allowed by the contractual framework they agreed on before to start the LB textbook. We expect modifications to existing content both because when we compose it with other material we need to make sure that the text flows properly and that terms are used consistently, but possibly also to add/edit content.

There may exist several versions of a LB textbook, per year, per professor and per class. Versions are different from editions: versions are different views on the LB content, i.e. the content can be assembled in novel ways depending on the needs of the specific audience: a book for a class of master students or PhD students, a book to improve personal knowledge of a topic, not for teaching purpose, etc. Instead, editions are "solidifications" of the LB textbook, whose content remains stable and fixed during the years. Note that while editions are static collections

generated from specific versions of the LB (they do not change anymore), the LB itself remains in its evolving state. As a result, authors can progressively improve and maintain the book, generating new editions as they reach a delta modification that they consider relevant.

This model is not that far from open source software or wiki-like projects, also here there are the concepts of collaborative efforts, sharing and reuse of materials, redistribution. However, rules and practices typical of wiki or open source projects cannot be straightforwardly applied to LB textbook scenario. The difference is that here authors have more and strong concerns about credits (intellectual property and royalties) or not proper reuse of existing work. Indeed, a distinction has to be done between "functional work", (e.g., manuals, computer programs, recipes) where it is somehow easier to establish rules about reuse and modification of contents and "opinion-type work" (e.g. scientific reports) where a modification can lead to misrepresent the authors' ideas or thoughts.

Summarizing, in our LiquidBook textbook scenario and pilot, our main objectives have been to explore:

- how to lower barriers and shorten time to market in the publication process of books;
- how to support a more agile development: up to date concurrent editions, tailored to readers;
- define novel credit attribution rules and less constraints in order to facilitate collaboration and reuse: editor(s) can take/edit/aggregate content from various contributors.

6.2.2 Conceptual Model

Figure 24 shows our current conceptual model for LiquidBook. In the model, we can see how the LB authors can start a new version, each composed by different types of content. Both structural and representation links are used to navigate the version. A structural link defines relations within the LB as an index while a representation link leads to the actual content in a particular representation.

The LB version has also one or many associated renderings (how the content is actually presented to the final readers). The evolution of a particular version can be tracked by temporal links. Every now and then a version is frozen in an edition. It is possible to define who (person) contributes to each version and what role (authors, contributors, etc.) this person takes on. People can annotate (with tags, comments, opinions) both content nodes and link nodes, and the system keeps track of who tags what. We can describe the working area of a LB as a "big pot", where authors put their content to share it with others and where they can define their personalized views (new versions) on top of this content. Versions are liquid, as they are in a continuous evolving state, while editions are versions, which have been solidified and cannot change anymore.

A version is a collection of multi-faceted content. The notion of multi-faceted content comes from the idea that for any given concept, the same concept can be "represented" in different ways: slides, text, video, etc. As an example, consider three professors teaching topics related to Web Applications in different universities. They decide to write an LB called Smart Web 2.0 Applications. They can have in a particular version some content related to REST, and different representations associated with this concept: slides, code libraries, text descriptions, as well as some content on Crawler, including just some text and a video of an interesting talk. The specific LB can then be

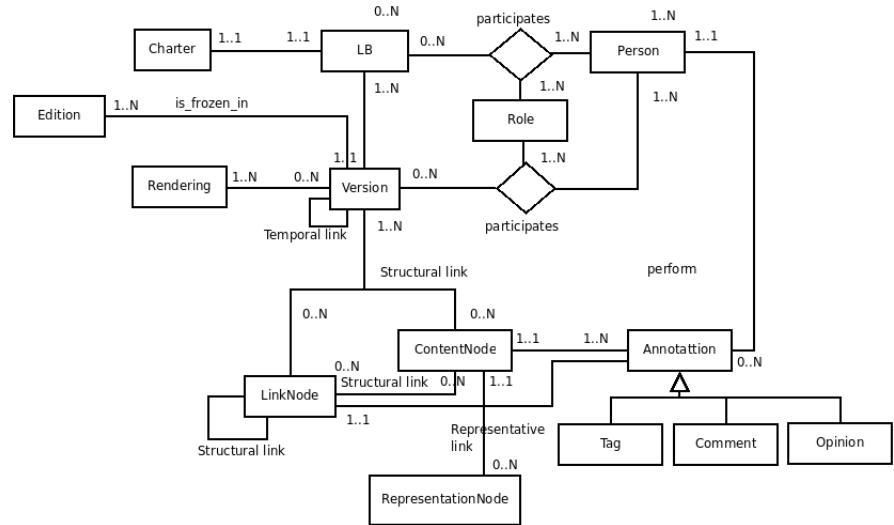


Figure 24: Liquid Book Conceptual Model

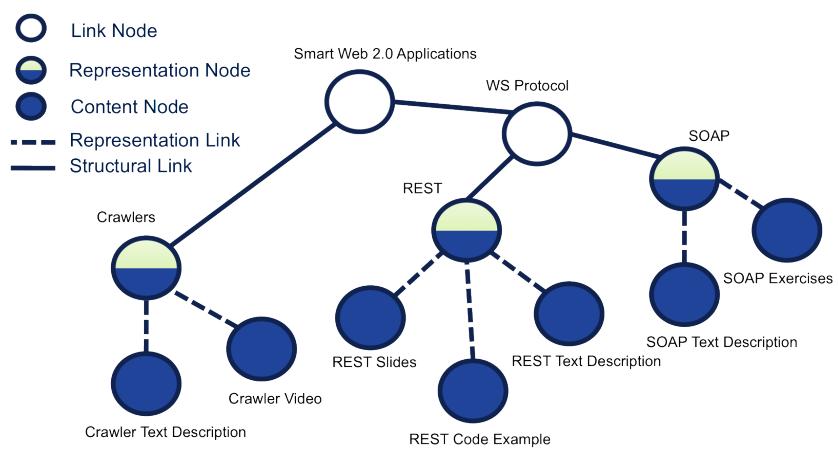


Figure 25: Example of LiquidBook structure

represented using a tree, where the nodes are the different topics (REST, Crawler, etc.) and the leaves the actual content (see Figure 3). Therefore, informally, we can distinguish three kinds of nodes in the tree: (i) nodes which just represent the different topics included in an LB, but do not contain actual content, like REST, SOAP, Crawler and that points to (ii) nodes where the actual content is (text, slides, code, etc.), and (iii) nodes which contain just a set of links that points to the different topics of the LB (WS protocol, etc.).

More formally, looking at Figure 25, it is possible to distinguish the following three kinds of nodes in the tree:

- a link node, which is a collection of links to other nodes, among the two in the figure we can distinguish the link node which is the tree root and identifies the particular version of the LB;
- a representation node, which is still a collection of links to different representations of the same content;
- a content node, which is the node containing the actual content. Link nodes and representation nodes are linked to each other through structural links, while content nodes are linked through representation links to content nodes.

As we mentioned before, LBs are evolving books, where versions, differently from editions, are not static collections, and are instead in constant development. To keep track of this evolution in time, we can use temporal links and series. Temporal links connect different trees of the same LB. Each tree is a version and the Series is the ordered list of tree's roots connected by temporal links, representing the evolution of the LB. A series keeps track of the evolution in the time of the LB version, where a new root node is created by the author(s) any time a significant delta of changes is reached.

6.2.3 Related work

Wikibooks¹⁴ (WB) are open-content and freely distributed, as they adopt a GFDL license¹⁵, which ensures that content remains copyrighted to the authors, while at the same time the copyleft licensing allows one to freely distribute and reproduce such a content. WBs are “public” in that everybody can see everything. Also, in WB anyone can contribute by adding new content, editing or deleting existing one. Everybody can participate, so a WB can have many authors. The main idea of Wikibooks follow the successful Wikipedia commons-based peer production model [6] incorporating three main concepts: (1) the content artifacts must be modular; (2) in order for a peer-production process successfully to pool a relatively large pool of contributors the modules should be predominantly fine-grained; and (3) the peer-production process must have low-cost integration - the mechanism by which the modules are integrated into a whole end product. Social norms play also an important role in sustaining some of these peer collaborations, both where there are small groups, and where there are larger groups and the platform allows for good monitoring and repair when individuals defect. At present, the Wikibooks site is listing 2,362 books in English in its repository and no specific ”best-seller” in the academic domain.

¹⁴<http://en.wikibooks.org/>

¹⁵<http://it.wikipedia.org/wiki/GFDL>

LB are different from the WB model in that:

1. LB are not necessarily open-content. The content is exposed through editions. This does not exclude (but also does not require) an approach where the content is also made public or freely available (based on certain licensing policies defined by the contributors or the publishers)..
2. The collaboration in LB is more on the contractual side: contributors may collaboratively edit a text, as done in wikis, but this may or may not happen and it is not the essence and value of LBs. The essence is in allowing reuse and modifications based on contractual agreements that define attribution of royalties.
3. Each piece of content in LB can exist in multiple versions, both because the original contributor can evolve it and because other contributors may take it and change it for putting it in their own edition. Around WB there is instead a social interest in having a *reference point*, a unique version which can be taken as reference.
4. In LB, the responsibility for the content ultimately rests on the LB editors of an edition. While in WB the focus is on the content creation process, in LB it is on the edition creation process by reusing, modifying, and organizing content. A distinction has to be done between “functional work”, (e.g., manuals, computer programs, recipes) where it is more easy to establish rules about reuse and modification of contents and “opinion-type work” (e.g. scientific reports) where a modification can lead to misrepresent the authors ideas or thoughts [28].
5. Editions are static collections (an edition is analogous to the edition of a book today), the LB itself remains in its evolving state and contributors can keep adding or editing content. We can see this as a branch in software development.
6. In LB we still have the notions of publishers and editions with ISBN, DOI etc.

With respect to other related efforts, a first attempt to create evolving textbooks is in place by MacMillan, one of the five largest publishers of trade books and textbooks. There are several points in common between our vision of LB and MacMillan's one of DynamicBooks¹⁶, as well as some differences. DynamicBooks will allow instructors to edit digital editions of textbooks. In this way, professors will be able to customize the book for the needs of their classes, deleting chapters, uploading different kinds of materials (notes, videos, pictures, graphs). However, the big breakthrough is the fact that professors will be completely free to rewrite or delete paragraphs, equations, illustrations without consulting the original authors or publisher, they have just to log into the system.

Regarding the business aspects, MacMillan will sell online access to the students, and the modifiable e-book editions will be much cheaper than the traditional printed version; even if the publisher will also offer print-on-demand versions of the customized books for a higher price. This can be seen as a particular case of liquid book with specific and very simple predefined rules and processes. They do not envision any customizable publishing process or legal framework to

¹⁶<http://dynamicbooks.com/>

constrain possible changes to books, publishing of editions, attribution of credits and royalties, and the like. On the contrary they rely completely on the instructors and on students or other instructors to help monitor changes, even if the publisher reserves the right to “remove anything that is considered offensive or plagiarism”.

For what concern the *collaborative* aspect, there are several examples of collaboratively-created books, even if they are more similar to classical e-books, without the multi-facet aspect or the evolving one. A first example is “*How to Think Like a Computer Scientist*¹⁷”, series of publications by Green Tea Press, where the same core programming text has been adapted to several different programming languages (Java, Python, C++, etc.). On the same line, *97 Things Every Programmer Should Know*¹⁸, another example of collaboratively written book, with hundreds of contributors, where multiple and varied perspectives are collected about things a programmer should know. Contributors have become involved through announcements, specific invitations, recommendations and word of mouth. Another example of collaborative written book is *Business Model Generation*¹⁹, with 470 co-authors and distributed without a publisher. The last three initiatives do not have a specific platform for collaboration, they usually rely on call for contributions and then on manual integration of contributions. A first attempt to facilitate the search of contributors and the collection of contributions is the one by Fast pencil²⁰, a platform to write, publish and sell collaboratively written e-books. The above mainly focus on collaboration on content creation, which again is not the focus of LB.

6.3 Instant Communities

The text of this subsection is reused from the Instant Communities (IC) demonstrator document [19] in order to keep this deliverable self-contained.

6.3.1 Introduction

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 we ever do that). There is often no easy way to follow-up with panelists.

The Instant Communities tool provides an IT infrastructure that helps create a “community of interest” in real-time during the panel or session. *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

¹⁷<http://openbookproject.net/thinkcs/>

¹⁸http://programmer.97things.oreilly.com/wiki/index.php/97_Things_Every_Programmer_Should_Know

¹⁹<http://www.businessmodelgeneration.com/>

²⁰<http://www.fastpencil.com/>

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.

Instant Communities is build on the notion of *Knowledge Spaces*, 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. In what follows we describe the main concepts, models and architecture of Instant Communities.

6.3.2 Conceptual model and architecture

Instant Communities is based on a set of models and IT infrastructure that provides the building blocks for the features and services provided by the tool. The models of Scientific Resources and Knowledge Spaces, described in the previous sections, provide these foundations. In fact, from the point of view of Knowledge Spaces, IC is seen as an application that extends from its concepts to support knowledge sharing in the context of panels, talks and other types of discussions

In Figure 26 we can see the SRS conceptual model with the configurations highlighted for entities, relations, annotations and resources.

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. 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 to denote that the information is related. For example, an attendee may upload a document or comment over the presentation of one of the panelist to denote

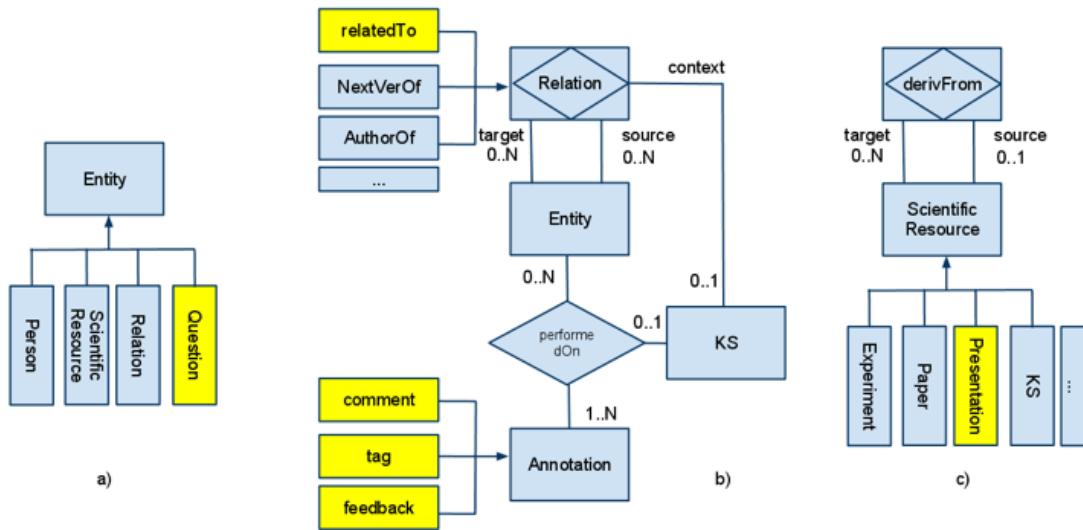


Figure 26: IC extensions to the SRS conceptual model in terms of a) entities, b) relations and annotations and c) resource types

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.

IC also defines some particular types of spaces such as *event*, *panel*, *topic* as well as high level roles such as *attendee*, *panelist*, *moderator*, *guest* given to the users participating in the panels. Figure 27 illustrates these configurations.

The architecture of the Instant Communities application is shown in Figure 28.

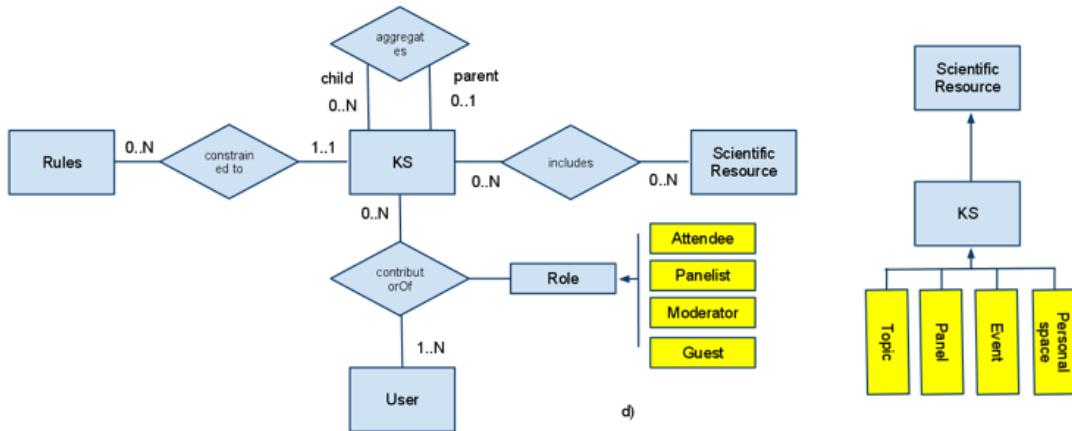


Figure 27: IC configuration on the KS conceptual model

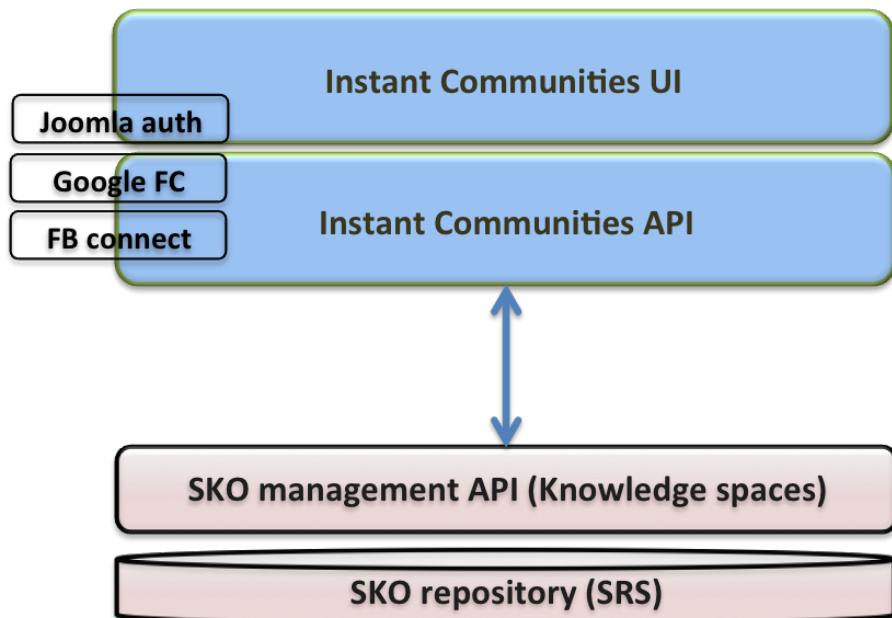


Figure 28: IC architecture

6.4 Liquid Journals

6.4.1 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. This is a big opportunity for making communication and knowledge transfer more efficient and almost real-time, but results in an information overload in terms of scientific content. Although we have a sea of information at our fingertips, it is not easy to find for interesting contributions.

An additional and somewhat puzzling problem 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 on peer review" in many different fields (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 TiVo²¹ or analogous digital video recorder makes a wide set of TV programs available to us so that we tend to watch what we know we like, and are less encouraged to look for new programs. The same effect has also been observed in science [25], 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 or CiteSeer. This is useful but it does not come near to solving the 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.

The analogy of Web pages points out to another limitation of the current dissemination model: while on the Web we have fairly established and accepted ways to *rank* resources and to *navigate* them, in science this is not always the case. There are many debates – and many papers - on which scientific metrics are "good", along with evidences that commonly adopted metrics (e.g., citation count) have many flaws. Furthermore, contributions other than papers are not typically evaluated, and the only "link" between scientific resources used in practice is that of citations.

Liquid Journals (LJ) are a way to overcome the information overload issue in scientific publications while also allowing to connect and collaboratively evaluate all kinds of scientific resources. Their underlying principles consist i) in leveraging the very same (large) community of scientist

²¹ www.tivo.com

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 computing scientific diversity and "enforcing" it when providing information.

LJs put this principles at work through concepts, methods, and ultimately tools. In a nutshell, the method is based on facilitating – and making it convenient – for scientists to share the implicit choices and observations they make every day in terms of *selecting* and *linking* knowledge. We all read papers and collect them, group them, share them via email with our colleagues or team (and therefore implicitly stating that it's worth to spend time reading them), "mentally" annotate them or observe that they build over this other paper or experiment or datasets, etc. If we could exploit this knowledge we would have a way to assess scientific resources of all kinds, and to link them, thereby facilitating evaluation, search and navigation. The "problem" is that today we don't bother to communicate this knowledge – or if we do, we do it verbally while talking at the coffee machine, or perhaps via email. We do not have time (or sufficient motivation) to go put this information somewhere to let the community benefit from this and probably we would not even know how to communicate all this. Hence, one of the main intuition – and challenges – we try to exploit is how to facilitate the above and making it convenient for scientists to implicitly or explicitly share this information and for a system to process it for providing the benefits as described above.

As discussed in the previous section, in the following we present the LJ model without referring to SKOs and SKO terminology. Later in this document we show how LJ concepts can be mapped into SKOs (and an adapter has also been built for this purpose). We now present the usage models and derived metrics along with a diversity-aware information prioritization model. We also present the architecture, implementation, and usage of the platform, which is available at liquidjournal.org

6.4.2 Scientific Resources for Liquid Journals

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, and is therefore essential to support the features discussed above.

LJs see scientific contributions as a structured, evolving, and multi-facet objects. Specifically, we see the space of scientific content we want to search, assess and disseminate as consisting of *scientific resources*, organized as set of nodes in a graph, that can be connected and annotated by authors 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), which go beyond what is currently captured today (which is essentially limited to authorship and citation).

We see these relationships as *subjective*, in that each person can in principle have different opinions on how two resources are related or even on the authorship (that is, on how or how much a person contributed to the knowledge represented by a scientific resource). So, in the LJ scientific resource model, we do not assume any universal truth. Relationships (including authorship), like annotations, are defined by a person or institution that therefore claims that relation to be true. For

example, I may state that I contributed to a paper. Relations can then be certified by certification agencies or endorsed by people. For example, Springer may certify that I, indeed, am author of a paper, and so that the relation is true.

In the current model we do not assume any method for ascertaining truth: we simply let users create and/or endorse relations, record them, and let the community manage and evolve them. This is analogous to what Wikipedia does and it works fine for that purpose. It is possible that this "democratic", honor approach over time does not prove to work and in that case a solution will have to be studied also depending on which specific aspect fails to be effective in the this community-managed approach.

The reason for allowing anybody to define relationships is because in this way we can leverage the power of the community to build contextual 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.

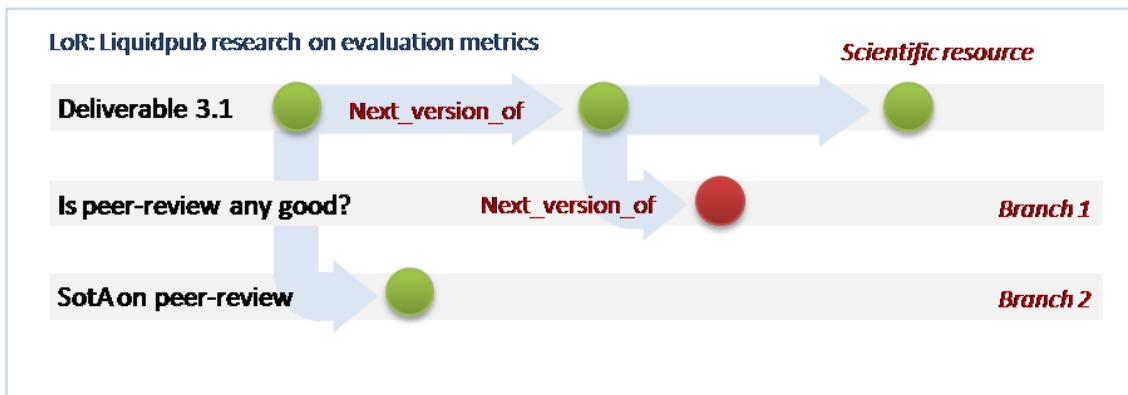
Figure 29 shows an example of resource graph, which illustrates relationships and their purpose. To illustrate these concepts, in Figure 29(a) we show the work of the LP research group on evaluation metrics and peer review. We started this line of research with in the context of a project deliverable (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, D3.1v1²² and D3.1v2²³, and we plan to produce in the near future a third 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. This is important as when we search for information, these links help us navigate to the latest version of a scientific contribution and it makes explicit the incremental nature of a result, thereby allowing reputation algorithms to factor this in when computing reputation of a scientific contribution. While working on this, we wanted to maintain the state of the art on peer review we had produced, and therefore, so we created a "spin-off" of the deliverable (D3.1v1) that we maintain in a distributed control version system²⁴. Analogously, 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?". These types of spin-off are 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 research result has many representations (Figure 29(b)). 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 29(c) we illustrate the use of particular dataset and experiments. The information described in the example captures the semantics of the relations, which today are hidden in the text and can be understood only after reading all the scientific resources.

Liquid Journals adopt the model of she space of scientific resources presented in Section 2. While the model allows anybody to create any kind of relation, the liquid journal model assumes

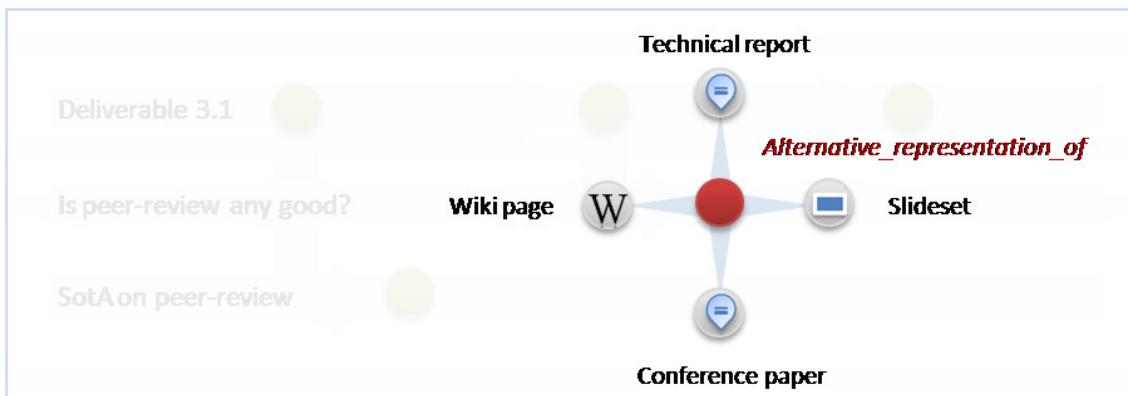
²² https://dev.liquidpub.org/svn/liquidpub/final/Year1/LP_D3.1.pdf, accessible using your reviewers' credentials

²³ <https://dev.liquidpub.org/svn/liquidpub/wp3/d3.1/v2/>, accessible using your reviewers' credentials

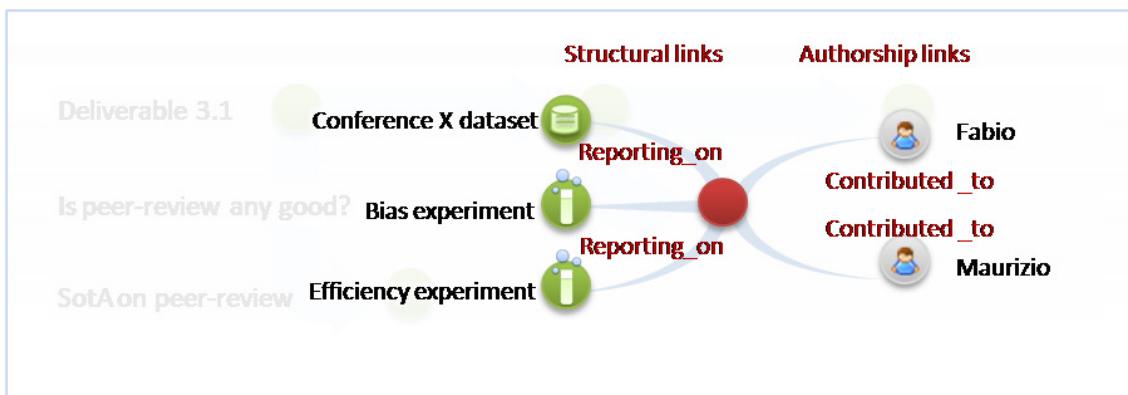
²⁴ <https://launchpad.net/liquidpub/peer>



(a) Example of scientific resources connected by next-version-of relation



(b) Example of different representation for the same resource



(c) Example of other general relations

Figure 29: Relations between resources

and leverages specific relation types, to which it assigns an agreed semantic (and also graphical interaction patterns in the Liquid Journal interface):

1. *Temporal relations* model the evolution of a contribution, 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 29(a) 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). In particular, LJ assumes resources to be organized in trees, where each tree essentially denotes a line of research and tracks the evolution in time of related documents. The tree is described by relationships of type $\langle \text{child next_version_of parent} \rangle$, and LJ assumes that each child has only one parent. This model for SR reduces overload because it clusters contributions into research lines (which are themselves SRs) and then allows users to navigate through contributions in the line of research. It also allows to more fairly attribute credit to contributions or authors by making explicit the incremental nature of a contribution
2. *Representation relations* model the multi-faceted nature of scientific resources. 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. In particular, LJ assumes that the relation type $\langle B \text{ alternative_representation_of } A \rangle$ denotes that resource B is a way to render/present the concepts/content of resource A (e.g., B can be a short version of a long paper A). To simplify the model and make visualization manageable, we assume that if B is a representation of another resource A, there can be no resource C who is representation of B. In other words, representations are arranged in a star where there is a resource at the center, and many alternative possible representations can be connected to it.
3. *Structural relations* represent arbitrary relationships between contributions, where the relationship may be further described by annotations. For example, a paper can be related to a dataset in that it describes results of experiments on that dataset. LJs exploit these structural relationships:
 - (a) $\langle \text{Re performed_on dataset Rd} \rangle$ denotes that Re is an experiment, Rd is a dataset, and Re is conducted on top of dataset Rd.
 - (b) $\langle R1 \text{ reporting_on } R2 \rangle$ denotes that R1 describes a dataset R2 or reports on the result of an experiment R2.
4. *Authorship* relations denote who contributed to the creation of the resource.
 - (a) $\langle P \text{ contributed_to } R \rangle$ indicates that person P contributed to the creation of the scientific resource R the typical example being the authoring of a paper. A typical annotation for this relation would denote the kind of contribution. For example, it may state that P contributed the state of the art section, or that P simply contributed by hiring the people who did the actual work.
 - (b) $\langle P \text{ edited } JI \rangle$ indicates that person P has been the editor of a journal issue JI.

5. *Usage* 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.

6.4.3 Liquid Journal Model

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 (web pages, traditional journals, etc). *Many journals can then refer to the very same scientific contribution.* This "appearance" of contributions in journals is an important information we exploit for capturing interest. In terms of the model described above, we represent journals as a collection of scientific resources (which are indeed links, as discussed earlier).

The resources that end up 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 on the reputation metrics and of the diversity-aware module as discussed next. 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. 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.

Formally, a liquid journal LJ is a specific type of scientific resource. Unlike a paper that proposes a new idea or a new interesting content, it is a resource whose scientific value is that of collecting, aggregating, and organizing content. As such it helps in disseminating scientific knowledge and it contributes to science. The essence of the journal is manifested via specific relationships with - and constraints on - other resources it is related to. A liquid journal LJ is characterized by (related to) a set of *editors*, one or more *publishers*, a default *editorial process*, and a set of journal *issues*. Issues are themselves resources, and can have editors and can be related to processes in case the selection and publishing process for a specific issue differs from the default one. Both the journal and the issues are then characterized by a set of scientific resources, that represent the content of the issue. In essence, the journal has content that evolves (possibly even continuously), while issues typically represent snapshots of a journal that the editor decides to "solidify". All these properties are expressed via distinguished relationships (such as "contains" or "is edited by") that are known to the LJ software. Notice that since resources are links, solidifying an issue means solidifying the links, not the content pointed by the links.

Because of the assumption on resources defined in the previous section, this means that a journal is now a "view" over resources available in the resource space, that can correspond to different kinds of contributions (or annotations over contributions), that can be navigated to find different versions or representations, and that can be more easily consumed and understood thanks to annotations and relations to comments, datasets, experiments, or blogs. The essence to make things work is that the relationships are maintained by the community and, as we will see, the members in the community do so based on their own selfish interest. We will also see that the liquid journal approach facilitates credit attribution, supports diversity and broadness in search, and encourage positive behaviors in science, and makes the scientific dissemination process more efficient. We begin by discussing how the LJ application supports the creation of liquid journals and showing how, while searching for interesting content to fill our journals, we provide an alternative and we believe more efficient and effective way to assign reputation to science and scientists.

6.4.4 Creating and Filling a Liquid Journal

A liquid journal can be created in two ways: by selecting and adding content manually or via a web search. In both cases, the characteristics of the LJ model are exploited to facilitate the collection and identification of the appropriate content to share.

Manual Content Selection. Manual selection mimics the way we collect and share interesting content today. There are various ways in which we identify scientific artifacts we'd like to read, use or share. For example, colleagues send us links or attachments via mail, recommending us to read them, and possibly even adding why (e.g., "take a look at this paper, it has a nice idea although it is not really developed or validated"). Or, we can stumble upon interesting material while searching or browsing the web. As another example, we may be listening to a talk at a conference and be browsing the conference proceedings and decide that the paper being presented is interesting, and that we should take the time to read it at the earliest opportunity.

The LJ infrastructure (and ecosystem, as we will see) supports this by facilitating editors in

1. **Gathering** scientific resources of interest into a *cooking area*, which is where resources are "parked" and then explored before being possibly published.
2. **Exploring** the relationships of the resources in the cooking area with other resources (e.g., newer versions of a paper, updated releases of a scientific experiment, different representations of a concept).
3. **Selecting** the actual resources we want to place in the journal
4. Creating a new issue and **publishing it**

The gathering phase can be done over time in one of the following ways: i) by dragging and dropping PDF files we have on our PC or receive as email attachment; ii) via "add to my journal" links that web site in the LJ ecosystem place next to scientific resources, much like links to digg and delicious; iii) via browser plugins that detect scientific resources in the page, analogously to what CiteULike plugins do for some web sites and some papers; iv) via pictures of papers, e.g.,

via an iPhone application that people can use to take a picture of a printed paper. For both the mobile phone example or the PDF email attachment, the LJ infrastructure takes care of identifying the link to the scientific resource available online and adds this link to the cooking area.

The exploration phase is performed by accessing the cooking area and exploring resources much like readers of a journal can do. In this exploration, editors can decide to exclude items from publication, or to take additional / alternative versions and representations of a resource, or other related resources that can be explored by navigating the resource graph.

The selected contributions can then be published in a new issue of a journal and hence shared with our team, our colleagues, or the entire community. At this time, because resources have been included in a journal, reputation is also assigned to these resources, as discussed next.

Content Selection based on Relevance, Interestingness and Diversity. Search-based selection differs from manual selection in the way the cooking area is filled up. With search, the cooking area is filled with items that are related to certain keywords or related to a resource or set of resources. For example, we may search for papers and datasets on peer reviews, or that are related to a given paper, or that use a given dataset. LJ search results are computed based on three dimensions: relevance, interestingness, and diversity. At this stage of the work we have not done proper research work in search over LJ. However, we describe some initial ideas and initial implementations, we discuss the possibilities that the LJ model offers in terms of search, and how they can be exploited.

Relevance can be computed with traditional techniques (for example, searching for keywords included in titles or text of papers) but in LJ can also leverage the links among resources as well as keyword search in annotations. In LJ we do not have the bandwidth (and probably the competences) to do research on keyword extraction or keyword search but we will rather reuse methods already available. What can also be done in LJ is to use the relationships among resources to provide search results, and in general we can use a Page Rank approach to rank results.

LJ supports search by interestingness by computing reputation metrics looking at how resources are “used” in journals (included in journals, or shared, or annotated). The detail of how this is done are described in the deliverables of WP 4, but we provide hints below that emphasize how LJ usage can support novel reputation metrics that we believe to be more significant for search and ranking than traditional ones.

Besides ranking by relevance and interestingness, LJ supports broadness of search in terms of searching for contributions coming from different scientific communities. We experienced first-hand the importance of this when searching for prior art on peer review. In this work we collaborated in a multidisciplinary group that included computer scientists, physicist, and philosophers. When looking for papers on this topic, somehow each group ended up finding content in its particular community of reference. Thus, for each group there was a lack of *diversity* in the discovery process. To avoid this problem LJ aims at enforcing diversity when providing search results, both to achieve a broader perspective on a topic and to achieve cross-fertilization between communities. A simple way to do this is to tag each resource with community information and have a graph of communities with measures of distance among them. We have done initial work in this direction by discovering communities in a bottom-up fashion, by looking at publication data. We have also discovered mappings between people and papers with communities. We refer readers to the CET

website²⁵ for further information on the ongoing work on community discovery. More sophisticated approaches can be developed by looking at the relationships among resources defined within the LJ application itself. This and other approaches (and tools) for enforcing diversity are part of our research agenda and are not described further here.

6.4.5 Usage model and evaluation metrics

A value proposition of LJ is that editors and readers provide knowledge that can help connect and assess scientific contributions. This happens in three 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 papers 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 initial 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 a resource to related resources and therefore to find “related” information.

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 do 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., who do so 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.

The conceptual model of LJ also provides the information to capture these aspects in the dimensions of the scientific resources, in the subscription links, in the structural links that make contributions appear in journals, in the usage information (tags, forwarding, sharing).

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. However, as it is unfeasible to provide a unique (and accepted) magic formula

²⁵ Community Engine Tool (CET): <http://project.liquidpub.org/research-areas/scientific-community>

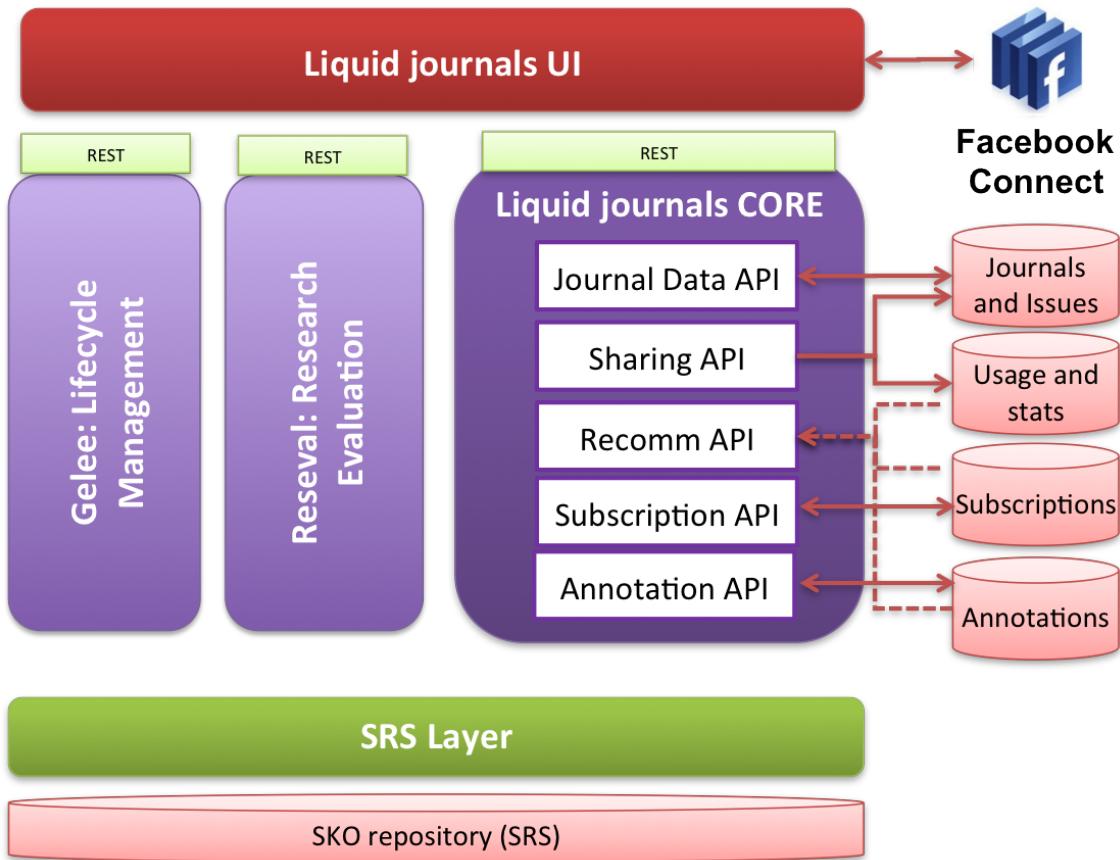


Figure 30: Liquid journals architecture

that captures all these aspects, in LJ we focus on providing the guidelines that will govern the instantiation of particular derived metrics. Indeed, we believe it is the community the one to decide what counts on the community. We are developing this concept with the metric *uCount*²⁶ that, as the name suggests, captures both the fact that everyone in the community counts and that everyone is involved in the process of defined 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. In LiquidPub we are also proposing a way to compute reputation out of opinions, which is embodied as algorithms and a tool, OpinioNet, which is already integrated to get information from LJ and to also provide reputation information back to LJ so that it can be used to rank search results.

6.4.6 Liquid Journal platform architecture

Designing and implementing an infrastructure for supporting the LJ model requires solutions and strategies for the different aspects of the model: i) managing the lifecycle of the journals, ii) journal creation, evolution, consumption and sharing; iii) access to scientific content in the Web,

²⁶In collaboration with ICST, icst.org

iv) computing the reputation of contribution (for ranking), and the v) projection of these features to a user interface. The liquid journals architecture relies on specialized components designed for each of the aspects mentioned. In Figure 30 we illustrate these components. Liquid journals provide a view of the scientific content available of the Web. As scientific contributions, in the broad meaning of the paper, fall outside traditional sources (e.g., digital libraries) where standards can be applied, the infrastructure requires an access layer that provides us the abstractions for easily accessing and searching content residing in non conventional, dispersed and heterogeneous sources on the Web. In order to address this requirement, we rely on the abstraction of Scientific Resource Space (SRS) [3] applied to the scientific domain.

On the foundation provided by the SRS, the liquid journal core component builds the services that support the LJ model. These services are organized in groups of APIs as shown in Figure 30.

As we need to provide LJ editors the possibility of defining their own journals' lifecycle, the architecture includes as lifecycle management component, the Gelee system. Then, OpinioNet computes reputation for resources based on the LJ data, as discussed later in this deliverable.

Services are very important in our architecture but to fully exploiting them it is necessary to provide a Web interface that facilitates the journal definition, search, content consumption and sharing. Without an appropriate interface, services are worthless. In this approach, we pay special attention to this issue and develop a rich Web application on top. We are also integrating our 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 service²⁷. At the time of writing, process and lifecycle management tools are not integrated with LJ as yet.

6.4.7 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 [27] and the overlay journal [24] introduce interesting ideas yet to be explored and taken beyond structural changes to meet the Web. These models (and the traditional model) 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., [18, 21]) 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 interest within communities. CiteULike²⁸, Mendeley²⁹, Zotero³⁰ and Connotea³¹ are examples of social bookmarking services with the focus on sharing and organizing academic references. These tools come with social tagging features that

²⁷ <http://developers.facebook.com/connect.php>

²⁸ <http://www.citeulike.org>

²⁹ <http://www.mendeley.com>

³⁰ <http://www.zotero.org>

³¹ <http://www.connotea.org>

allow people to collaboratively tag content so it can be easily found later. Thus, these tools provide storing, sharing and tagging of references to publications via shared collections and groups. In the case of public collections, readers can consume the content by browsing or via RSS feeds.

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 [22]. Specialized search engines, such as Google Scholar³² and CiteSeer³³, have been developed for searching papers/books across multiple repositories using crawling techniques and protocols. Using another approach, the academic search engine, BASE³⁴, 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 has also been used for bringing users content they might like [26]. This has led to general relevance and diversity algorithms proposals trying to balance user preferences and diversity (e.g. [17]). In the academic domain, recommendation of papers have also been explored in many studies (e.g., an evaluation in [23]).

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. Search by similarity is not provided (or limited) and diversity in the results is not enforced. 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 define the notions of interestingness, of diversity, of quality applied to scientific content, and combine them to rank and group scientific contributions.

6.4.8 Liquid Journals and LiquidPub

In addition to the benefits discussed in this section, with respect to the specific objectives of LiquidPub, LJ if successful and adopted as a model and system helps in achieving the following:

- it allows to model evolution and it facilitates people in stating evolution relations among resources by enabling authors, editors, and readers to define the relationships (and also annotate them to detail what the changes are about);
- it provides the basis for reputation metrics that are based on opinion of each of us in the scientific community by attempting to capture our implicit opinions we express by selecting

³² <http://scholar.google.com>

³³ <http://citeseerx.ist.psu.edu>

³⁴ <http://www.base-search.net>

and sharing content with colleagues. Furthermore, because incremental contributions are marked as such, it enables a proper evaluation of such increments (or of an entire line of research);

- it supports multi-faceting by considering resources of all types as first class objects, not just papers. It also makes it easy to navigate to them in searching for content or in reading a journal, as they are linked from other resources;
- it supports an efficient dissemination of knowledge, since posted contributions are spread by the community via a Web 2.0 version of the word of mouth. Contributions can be disseminated immediately when the authors think they are ready to do so and it can be done in the format and representation the authors think it is best for the specific content (or in multiple forms);
- because readers are likely to subscribe to content from journals and authors that post innovative ideas, early release of knowledge is more likely to be rewarded, and as such encouraged.

6.5 Liquid Museums

Note that this section tells about an application which we only start to develop, therefore, here you will not find architecture pictures or API descriptions. However, we keep the description to show how other initiatives can base on LiquidPub results.

Liquid Museums application that will be developed together with the Museum of Archaeology and Anthropology (MAA), University of Cambridge. MAA is planning a major redesign of their permanent exhibition to be completed and opened to the general public in early 2014. Intermediate events will be held between fall 2011 and 2014 for limited areas or targeted to specific audiences. MAA is interested in piloting a dedicated version of knowledge spaces technology to create a social and digital experience surrounding the visit to the museum and continuing after it. The idea is that an object in an exhibition is not simply enjoyed via its physical representation, but also through its history of events, through the research done related to it over the years, through the related comments and thoughts of scientists, and through the community of people interested in it.

Content related to the object may also be of general interest and provided either by visitors, or by professional media/content provider (e.g., pushed by BBC). The object may be “consumed” partly during the visit but then also once back at home or in office and the content of interest can be captured and shared with friends and colleagues. The consumption of the object is tailored to the type of visitor, possibly via a storytelling, but also via search of related interesting content.

The pilot will start in May 2011 with the first prototype going live in fall 2011. We are planning to use SRS and Knowledge Spaces to support annotation and interaction with objects.

References

- [1] ARCOS, J. L., ESTEVA, M., NORIEGA, P., RODRIGUEZ-AGUILAR, J. A., AND SIERRA, C. *An integrated developing environment for electronic institutions*. Birkhäuser Publisher, 2005, pp. 121 – 142.

-
- [2] BAEZ, M., BIRUKOU, A., CASATI, F., AND MARCHESE, M. Addressing information overload in the scientific community. *IEEE Internet Computing* 14 (2010), 31–38.
 - [3] BAEZ, M., AND CASATI, F. Resource Space Management Systems. In *Proceedings of the European Conference on Web Services (ECOWS 2009)* (2009).
 - [4] BAEZ, M., CASATI, F., AND MARCHESE, M. Universal resource lifecycle management. In *Data Engineering, 2009. ICDE '09. IEEE 25th International Conference on* (2009), pp. 1741–1748.
 - [5] BÁEZ, M., PARRA, C., CASATI, F., MARCHESE, M., DANIEL, F., DI MEO, K., ZOBELE, S., MENAPACE, C., AND VALERI, B. Gelee: Cooperative lifecycle management for (composite) artifacts. In *ICSOC/ServiceWave* (2009), pp. 645–646.
 - [6] BENKLER, Y., AND NISSENBAUM, H. Commons-based Peer Production and Virtue*. *Journal of Political Philosophy* 14, 4 (2006), 394–419.
 - [7] CASATI, R., ORIGGI, G., AND SIMON, J. Is this the scientific article of tomorrow? (submitted). *submitted to Journal of Documentation* (2010).
 - [8] D2.1v2 process, role and licensing models for sko lifecycle management, 2011.
 - [9] D2.3v2 liquidpub processes plug-ins (prototype), 2011.
 - [10] D3.2 model of reviewers' behavior in peer reviews, 2010.
 - [11] D3.3 simulation and validation of the behavioral models, 2010.
 - [12] D4.1 credit attribution for liquid publications, 2010.
 - [13] D4.2 co-determination models of reviews, 2011.
 - [14] D4.3v2 development of the analysis and evaluation plug-ins (prototype), 2011.
 - [15] D5.2v2 integration of plugins, 2011.
 - [16] ESTEVA, M. *Electronic institutions. from specification to development*. PhD thesis, Universitat Politecnica de Catalunya, 2003.
 - [17] HADJIELEFTHERIOU, M., AND TSOTRAS, V. Optimization of multi-domain queries on the web. special issue on result diversity. *IEEE Data Engineering Bulletin* 32, 4 (2009).
 - [18] HIRSCH, J. 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 (2005), 16569.
 - [19] Instant communities. demonstrator description, 2011.
 - [20] Liquid books. demonstrator description, 2011.
 - [21] KRAPIVIN, M., MARCHESE, M., AND CASATI, F. Exploring and Understanding Citation-based Scientific Metrics. *Advances in Complex Systems* 13, 1 (2010), 59–81.
 - [22] LOSSAU, N. Search engine technology and digital libraries. *D-Lib Magazine* 10, 6 (2004).

- [23] PARRA, D., AND BRUSILOVSKY, P. Evaluation of Collaborative Filtering Algorithms for Recommending Articles on CiteULike. In *Workshop “Web 3.0: Merging Semantic Web and Social Web”* (2009).
- [24] PINFIELD, S. Journals and repositories: an evolving relationship? *Learned Publishing* 22, 3 (2009), 165–75.
- [25] SANDWEISS, J. Essay: The Future of Scientific Publishing. *Physical review letters* 102, 19 (2009), 190001.
- [26] SCHAFER, J., FRANKOWSKI, D., HERLOCKER, J., AND SEN, S. Collaborative filtering recommender systems. *The Adaptive Web* (2007), 291–324.
- [27] SMITH, J. The deconstructed journal—a new model for academic publishing. *Learned Publishing* 12 (1999), 79–91.
- [28] STALLMAN, R. M., LESSIG, L., AND GAY, J. *Free Software Free Society: selected essays of Richard M. Stallman*. GNU Press, 2002.