D5.2v1
Integration of Plugins

<table>
<thead>
<tr>
<th>Maintainer/Editor-in-chief</th>
<th>Fabio Casati</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core authors</td>
<td>Fabio Casati, Cristhian Parra, Aalam Wassef</td>
</tr>
<tr>
<td>Maintainers/Editors</td>
<td>Aliaksandr Birukou, Maurizio Marchese</td>
</tr>
<tr>
<td>LiquidPub research group leaders</td>
<td>Fabio Casati, Roberto Casati, Ralf Gerstner, Fausto Giunchiglia, Maurizio Marchese, Gloria Origgi, Alessandro Rossi, Carles Sierra, Yi-Cheng Zhang</td>
</tr>
<tr>
<td>LiquidPub project leader</td>
<td>Fabio Casati</td>
</tr>
</tbody>
</table>

Grant agreement no. 213360
Project acronym LiquidPublication
Version v1.0
Date June 11, 2010
State Solid
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 deliverable describes how the services of the LiquidPub platform are integrated. Keyword list: integration, services, plugins, LiquidPub platform
## Contents

1 Introduction 1

2 Liquid Journals 5
   2.1 Source Code 5
   2.2 Current status 5
   2.3 Videos of demos 6
   2.4 Architecture 6
   2.5 API 6
   2.6 Other documents 6

3 Liquid Conferences/Interdisciplines 6
   3.1 Source Code 6
   3.2 Current status 7
   3.3 Videos of demos 7
   3.4 Architecture 7
   3.5 API 8

4 Karaku 8
   4.1 Source Code 8
   4.2 Current status 8
   4.3 Architecture 8
   4.4 API 8
1 Introduction

As in all prototype deliverables, this document serves simply as a pointer to the software artifacts, along with a brief illustrations of what they represent.

This document is a prototype deliverable. Before pointing to the software artifacts, we recap the LP architecture (to make this document self-contained and provide a proper reading guide) and also report on the status of the integration. We begin by describing the LP high-level architecture, with text taken from D5.1v2 [1].

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 way in which the LiquidPub platform (Figure 1) supports the LP vision is the following:

- Three pillars embody the LiquidPub principles in the three main ways scientific dissemination is achieved today: conferences, journals, and books. These are the metaphors through which we achieve the LP objectives with different dissemination paradigms. Within the LP...
platform, these are the vertical rectangles in Figure[1]

- A SKO knowledge repository and semantic knowledge bus integrates information across (and allows communication among) the different pillars, and also acts as an integrated repository of liquid knowledge.

- Reputation modules (OpinioNet and ResEval) process the information in the SKO repository 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 good for science.

- Tools for workflows and lifecycle management (LPMaSys 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 modules and the knowledge bus.

- Infrastructure services (Karaku) connect LP with existing sources of information and in general with the non-LP world. An example of this consists in exposing applications such as Google Scholar and Google docs via an API to the LP platform so that information from them can be integrated into the SKO repository or the pillars, or used by the reputation tools.

In terms of implementation strategy, while advancing with the design and implementation of the pillars, we progressively understood that:

1. The various LP pillars that deliver (along with the other LiquidPub components) functionality to users and ultimately deliver the LP vision will evolve over time also during the life of the project, especially as they become adopted and we learn how people use them. We feel this evolution is natural as the concepts are novel. As we dive deep into development and as we have early prototypes available, we better learn and understand requirements as well as how to refine, extend, and in general evolve the basic abstractions and underlying conceptual models. We cannot imagine to get things right the first time around.

2. Researchers in this area 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). For example, some researchers outside LiquidPub (from UNSW Sydney) are developing 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 Share[2] system from TU Eindhoven). Incidentally, 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 tightly couple among them the abstractions of each of the pillar from the very start. In fact, both from a conceptual perspective and in terms of implementation, in LP we proceeded by developing the three dissemination paradigms for the pillars. 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 will have the notions of conference, of paper to be discussed, of comments, etc. Liquid Journals will 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 will have the books and editions as first class objects. 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 model and, more in general, the functionality offered by each of the three pillars will evolve. 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 pillar 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 pillars so that they can share knowledge and put the functionality of one at the service of the other. For example, a scientific resource in a liquid journal can become a paper in a liquid conference that is the basis for a conference/discussion session, or vice versa. Or it can be taken as part of a liquid book.

The second implication is that we need to provide the hooks for other researchers to build over our results and easily integrate their idea with LiquidPub - as we certainly do not claim to have the copyright on all ideas in the area. We envision different kind of extensions, some will be directly extending the pillars, but others can build directly on top of the knowledge bus and of other “knowledge services”, as discussed below. This is what we mean by the “ecosystem”: providing a way for other R&D effort to integrate with LiquidPub (this is graphically represented by the xyz pillar in Figure 1), 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 three pillars.

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. In the knowledge bus, abstractions related to scientific knowledge are integrated and can be transferred across pillars. 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 pillars (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.

The SKO knowledge bus and repository also provide a hook for defining and developing one of the most important aspect of LiquidPub, which is reputation. Indeed, while a reputation module
could be developed for each pillar - more tailored but with i) more development effort as it needs to be developed for each of them, and ii) restricted in visibility to what is managed by that pillar - one option LiquidPub is pursuing is to achieve integration at the knowledge bus/shared model level. This means that reputation in LP entails mapping reputation-relevant information from the pillars 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 pillars initially have their own abstraction and conceptual model. These will also be the initial requirements for a knowledge bus/shared model.

- Because these abstractions and models will evolve over time (possibly in an agile and rapid fashion, especially in an initial period) we need to have a way to, on the one hand, let these models evolve freely (this is important to be able to capture and model the concepts that prove to be important), but on the other hand to consider that these changes do impact in some way the shared SKO model, that should indeed reflect them.

- Therefore, initially each pillar should have its own model and API, and possibly initially even its own physical data repository, which is required to support the fast evolution of each pillar.

- However each of them needs to be integrated, for the reason stated above. This observation has two sides: i) concepts and API of the pillars need to be mapped to concepts and API of a shared model/bus, and ii) over time the shared model should evolve (though, realistically, more slowly than the pillars) to accommodate new requirements at the appropriate level of abstraction. In other words, in some cases new concepts in the pillar 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 so that they are made available at the platform level.

According to the same philosophy, the infrastructure services (Karaku) can be queried from - and feed information to - the SKO repository as well as directly into the pillars.

This strong emphasis on “use cases” (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, and so the pillars are not described in other prototype deliverables. Therefore, we include here pointers to the prototypes (for journal and conferences, since books are not implemented). For completeness we also include below a description of the Karaku prototype, even if we do not see it as a key to prove the LiquidPub concepts (it is a nice add-on to facilitate integration with existing sources: the extent to which we will develop it further depends on how we will progress on the core LP components, the pillars, the SKO, and the reputation tools). The other software artifacts are discussed in the other prototype deliverables, namely: D1.3v1 [2] for SKO, D2.3v1 [3] for lifecycle management, D4.3v1 [4] for the reputation tools. 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.
We now report on the status of the integration among the different components. As we will see, in this version 1 of the deliverable we have achieved varying degrees of integration for the different components, with the goal of completing the integration for version 2 as components become more stable and understood. The current integration status is as follows:

- the Liquid Journal platform is integrated with the SKO platform (Bus and Repository) via an adapter that ports LJ content into the SKO platform and makes it therefore available to all modules that plugs on the SKO platform
- the Liquid Conference platform is conceptually integrated with the SKO, meaning that currently we have a specification of the behavior of the adapter between conferences and SKO, but not yet an implementation.
- ResEval is integrated with Karaku, meaning that it uses Karaku to access sources of publications, such as Google Scholar, and computing citation statistics.
- OpinioNet uses ResEval to gather basic citation data, and is integrated with the SKO module. What this mean is that currently we can compute reputation metrics based on usages of knowledge artifacts in the Liquid Journal platform.
- Gelee is integrated with Karaku in that it can model and execute a lifecycle over resources accessed via Karaku

2 Liquid Journals

2.1 Source Code

Liquid journal is released as an open source project under the GNU Affero General Public License (AGPLv3)[2] In order to actually reach the open source community, we made the application available at the well known open source community, Launchpad, and accessible at [http://launchpad.net/liquidjournal](http://launchpad.net/liquidjournal) However, at the moment, the source code is available at [https://dev.liquidpub.org/svn/liquidpub/prototype/liquidjournals/](https://dev.liquidpub.org/svn/liquidpub/prototype/liquidjournals/) (using lp-guest and lp-password credentials).

The code is organized in two separate (set of) components: the service side which implements the liquid journal features and expose them as RESTful services, and a set of client applications (web, mobile and browser plugins) that offers users the interface for interacting with liquid journals.

2.2 Current status

We have implemented the services for creating, filling and browsing liquid journals as well as the features for sharing, searching, annotating, linking and navigating scientific resources. All these services are provided by a backend application available at [http://liquidjournal.org/api/](http://liquidjournal.org/api/) On top of these services we have developed:
• a first prototype of the Web UI that implements the user interface for the features provided by the backend,
• an iPhone application that puts liquid journals into the most widely used phone in the market,
• an email interface that allows us to push all the data that is usually shared with colleagues by email, into the liquid journal platform, and
• a browser plugin to collect scientific resources and to fill liquid journals while browsing the web.

2.3 Videos of demos

You can find a video of how Liquid Journals work at http://liquidjournal.org/demo.html along with links to some live examples.

2.4 Architecture

A detailed description of the architecture of the Liquid Journals can be found in [1].

2.5 API

The information for developers is available at https://launchpad.net/liquidjournal where we provide links to the project wiki, homepage and other resources. In particular, the API specification is available at http://liquidjournal.org/api/.

2.6 Other documents

More information about Liquid Journals, such as presentations and related documents, is available at http://project.liquidpub.org/research-areas/liquid-journal.

3 Liquid Conferences/Interdisciplines

Interdisciplines is an online platform allowing the management of text-based conferences. It implements the Liquid Conferences use case and is available at http://www.collective-developments.org/interdis/index.php (till August 2010) and at http://www.interdisciplines.org (from August 2010).

3.1 Source Code

The source code of Interdisciplines will be released in autumn 2010.

---

3 For this and following links to Interdisciplines you must login with the following username and password: lp-reviewer lp@disireview
3.2 Current status

Interdisciplines now supports main features of the Liquid Conferences use case: modular paper structure with authors specified for each section, managing bibliographies, managing paper licenses, reviewing and commenting papers. It will be used for organizing a conference on copyright and licensing issues related to Scientific Publishing 3.0 in the summer 2010.

Future extensions of the platform include login with Academia.edu account and managing paper bibliographies via Mendeley and sharing them.

3.3 Videos of demos


3.4 Architecture

Interdisciplines’ architecture is composed of a backend and a frontend (public website). The backend relies on a relational database whose simplified overview can be seen in Figure 2. Following the same link, one will also be introduced to Interdisciplines’s UI for registered authors. The presentation at [http://www.slideshare.net/ColDev/interdisciplines-20](http://www.slideshare.net/ColDev/interdisciplines-20) provides a brief overview of Interdisciplines.
3.5 API

The API is not yet available. In the future, APIs for exchanging bibliographies, providing article metadata for liquid journals, and displaying reputation of articles, based on their inclusion in liquid journals might be provided.

4 Karaku

Karaku is a tool that connects the LiquidPub platform with existing sources of information about scientific contributions, researchers, events, etc. Resman is a component of Karaku that deals with information extraction from each information sources (SpringerLink, DBLP, Google Scholar, etc.).

4.1 Source Code

Both Karaku and Resman are released as open source projects under the GNU Affero General Public License (AGPLv3). The source code of Karaku is available (using lp-guest and lp-password credentials) at https://dev.liquidpub.org/svn/liquidpub/prototype/lpbase/ while Resman is available at https://dev.liquidpub.org/svn/holms/trunk/resman/.

4.2 Current status

We have implemented the services for creating, filling and browsing a simple and extensible model of scientific entities as well as the features for annotating, linking and navigating these resources. All these services are provided by a backend application available at http://project.liquidpub.org/karaku/.

As core layer of KARAKU, Resman fully supports navigation of adapters and web resource definitions plus execution of actions over resources for which adapters have been registered. Currently, adapters for the following services have been developed: GoogleDocs, Google Scholar, MediaWiki and Subversion repositories.

4.3 Architecture

The description of the architecture of KARAKU is described is D5.1v2.

4.4 API

The information for developers is available at http://docs.google.com/View?id=dg4fx7fj2fjw6pmc4 where we provide links to the project wiki, homepage and other resources. In particular, the API specification is available at http://docs.google.com/View?id=ddf93b973gfmdmf7c.

*Karaku is a Guarani word traditionally used to refer to the core of an issue
*http://www.gnu.org/licenses/agpl-3.0.html
References


