DataConf and its Linked Open Data ecosystem: produce, link and consume scientific conference metadata

Lionel Médini
Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205
F-69622, France
+33 4 72 43 16 36
lionel.medini -at- liris.cnrs.fr

Florian Bacle
Université de Lyon
Université Lyon 1
F-69622, France
florian.bacle -at- etu.univ-lyon1.fr

Fiona Le Peutrec
Université de Lyon
Université Lyon 1
F-69622, France
fiona.le-peutrec -at- etu.univ-lyon1.fr

Benoît Durant de la Pastellière
Université de Lyon
Université Lyon 1
F-69622, France
benoit.durant-de-la-pastelliere -at- etu.univ-lyon1.fr

ABSTRACT
DataConf is a mobile Web mashup application that allows browsing conference metadata (publications, authors, authors' organizations), as well as the conference schedule (tracks, sessions, talks). It uses Linked Data and Web APIs to enrich the conference dataset with resources (authors' other publications, organizations homepages) from different other endpoints. It also performs client-side reasoning to retrieve and recommend publications. Unlike other mashups, DataConf dynamically aggregates data on the client side. Server-side calculations are devoted to the queried data sources, not to the DataConf server itself. DataConf is easily configurable and deployable for any conference with available metadata on a SPARQL endpoint. Its component-based architecture allows developing new extensions to query extra data sources. We also propose several custom data sources, among which SimpleSchedule that provides a convenient interface for managing conference events and exposes them on a SPARQL endpoint and DataPaper that allows conference participants to enrich their own data.

Categories and Subject Descriptors
E.1.2 [**DATA**]: Data structures – *Distributed data structures.*

General Terms
Design, Standardization.

Keywords
mobile Web mashup, mashup architecture, mobile reasoning, Linked Data, Linked Data sources.

1. INTRODUCTION
The WWW'2012 conference that was held in Lyon was an occasion for the local Web community to initiate several innovating projects around conference material and Web technologies. We designed a mobile Web application that allowed the conference attendees to navigate among publication metadata (title, authors, abstract, keywords…), as well as authors and other publications metadata, using their smartphones and tablets.

As it targeted a WWW series conference, technical choices were oriented towards a full client-side mashup application applying recent/emerging Web technologies and standards (in particular, HTML5 APIs). It also takes advantage of Linked Data [1] by dynamically enriching publication metadata from several sources. We thus explored the feasibility, using currently available browsers on mobile devices, of: (i) dynamically constructing complex SPARQL queries and sending them to cross-domain endpoints, (ii) representing and allowing browsing among these metadata using a textual and a graphical interface, (iii) locally building, classifying and querying an ontology and (iv) capturing and processing images in JavaScript using the device built-in camera.

This app has been refactored to be more generic and several instances have been created for different conferences. It is now named DataConf [2] and provides a convenient means for conference attendees to access a conference program and schedule. Several DataConf instances related to various conferences are available at http://dataconf.liris.cnrs.fr/ and the sources are downloadable at https://github.com/ucbl/DataConf.

In this paper, we present a generic, configurable version of DataConf that queries different publicly available datasources from the LOD and Web API cloud. We enumerate the public datasources it currently relies on and describe with more details custom-made datasources that aim at managing several kinds of data related to the scientific conference domain. We also describe the underlying generic architecture that can be reused to create new single-page, client-side mashup applications able to query multiple datasources and to perform just-in-time processing of the retrieved data, eventually including semantic reasoning.

This paper is organized as follows: it presents the main functions of the DataConf webapp, the custom datasources we developed to enrich and publicize the conference datasets and a generic view of our client-side mashup architecture. We then conclude and present the evolution perspectives of this work.

2. BROWSING PUBLICATION METADATA

DataConf aims at searching, browsing and enriching scientific conference metadata. From the application homepage, users can access publication views (concerning papers, posters, demos, etc.) by author, keyword, title, session type (track, keynote…) or time schedule, as shown in Figure 1. The following sub-section describes how external resources are used to provide the conference metadata and enrich those data using external resources. The next subsection focuses on how DataConf uses them for organizing the data and presenting them to the user. Subsection 2.3 presents other internal functions of the application.

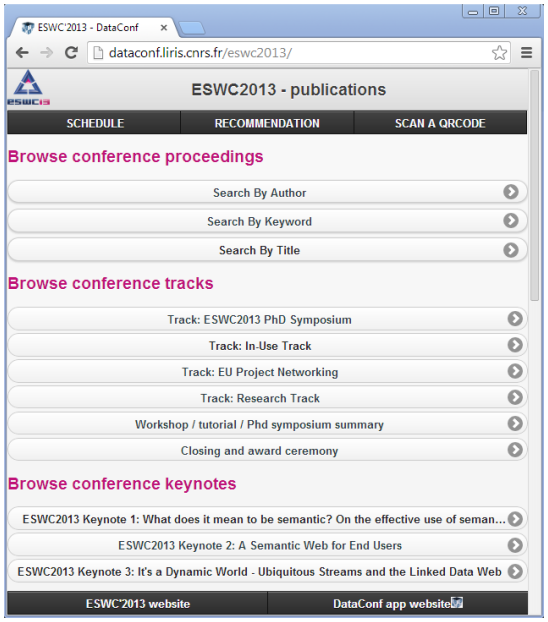

Figure 1. Homepage of the ESWC'2013 DataConf instance.

2.1 External Datasources

The **primary datasource** is a SPARQL endpoint exposing the conference metadata. Since 2006, conference publication datasets are available on the Semantic Web Dog Food (SWDF) SPARQL endpoint[1] [3]. These datasets mainly rely on the SWC vocabulary [4], thus integrating DC [5], FOAF [6], SWRC or ICal [7] elements. Such datasets contain information about the publications, their authors, but also the events that take place during the conference (tracks, sessions…) and the conference schedule. The original purpose of DataConf was to display information about publications (Figure 2a), authors (Figure 2b), organizations (Figure 2c) and keywords (Figure 2d), and allow the users navigate among the corresponding views. Originally, we expected the primary datasources to be available on the SWDF endpoint that contains (among other conferences) the metadata about the WWW'2012 conference in the SWC format. As it experienced availability issues while we were preparing the WWW'2012 and ESWC'2013 instances, we set up our own endpoint for accessing these two conferences metadata. Therefore, it remains possible to change the DataConf instances configuration to query another endpoint instead of ours.

DataConf enriches these data by querying other available datasources:

[1] http://data.semanticweb.org/

DBLP (DataBase systems and Logic Programming) [8] is a wider dataset about computer science publications. It contains specific metadata about the publications, such as title, DOI URI, year, publication type and name of the conference / journal. It is used to enrich the authors' views with their other publications, external to the targeted conference. The DBLP database is queried in Linked Data through the L3S[2] SPARQL endpoint. Nevertheless, DBLP has one drawback: it lacks publications keywords.

DuckDuckGo![3] is an online search engine. We use its Web API for enriching organizations' information, such as homepage URL and logo. This engine is queried using the organization full name. We rely on its "I'm feeling ducky!" feature to retrieve the most probable query result. We noticed that DDG provides quite good results for organizations, but less good results for people. Indeed, we originally used DuckDuckGo! for retrieving authors' homepages, but switched to Google while finding too many inaccurate results on this particular query type.

Google Web search API[4] is used to enrich data by finding authors' homepages, as explained previously. Homepages are displayed in the "Personal Page" section of the authors' views, see Figure 2b.

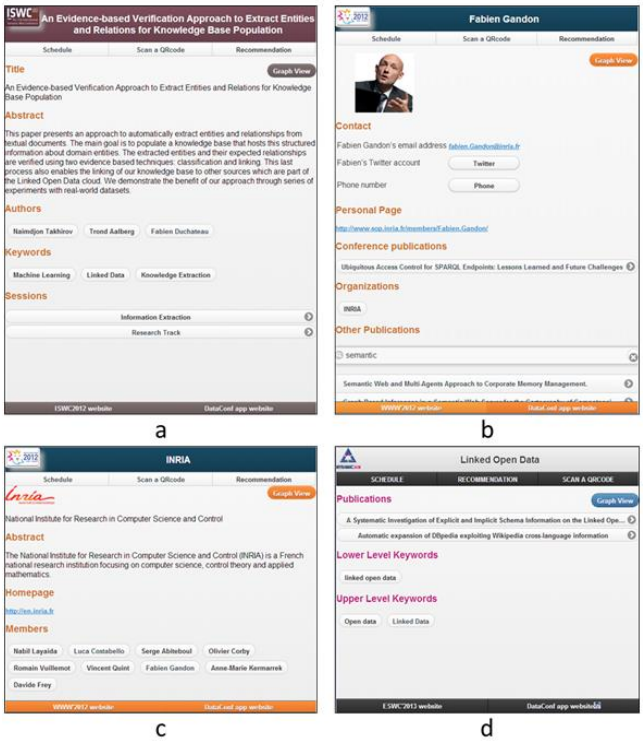

Figure 2. Different views in different DataConf instances: a) publication, b) author, c) organization, d) keyword.

At this point, the reader should note that DataConf did not yet retrieve some elements of the above views, such as author's photo, contact information and keyword hierarchies. These data come from the custom datasources and local reasoning described in the next sections.

[2] http://dblp.l3s.de/d2r/

[3] http://duckduckgo.com/

[4] https://developers.google.com/web-search/

2.2 Linked Data aggregation and browsing

DataConf thus constructs and sends queries to the available datasources, depending on the type of the RDF resource (publication, author…) requested by the user. It may store, display the retrieved data and even reuse it for sending further queries, as explained in section 6. The pieces of data that are determined to be displayed – according to a predefined template – compose the resource view. As most of them denote other resources, they are represented as navigable in the DataConf interface, in order to materialize the navigation process among the available Linked Data.

For instance, an author's view provides information such as name, most probable homepage, affiliation organization, publications in the desired conference and other publications from DBLP. Among these data, the homepage, organization and publication items are navigable. If the user clicks on the organization name, DataConf queries the main, DDG and Google datasources to display the logo, name and homepage of the organization, as well as other authors referred in the main dataset for this organization. If he/she clicks on another author's name, DataConf build a view similar to the preceding one, which describes this second author. Then, if he/she clicks on one this author's other publications, which data come from the DBLP datasource, DataConf displays a new view showing specific external publication metadata (see above), among which this publication authors. Of course, these authors are as well navigable.

Therefore, users can also use DataConf to browse the DBLP datasource, instead of the main conference one. This is a typical Linked Data usage example of the available data. Such a feature seemed interesting for evaluating the interest conference attendees would find in our interface for browsing large publications databases.

2.3 Other internal features

Publication or author search: several views provide forms that allow the user to type a publication title or author's name (see Figure 2b). In each form, automatic suggestions are provided by sending asynchronous SPARQL requests to the metadata server.

View metadata as graphs: users can choose to view and navigate in tree-based graphical representations of the publications, authors or keywords. The graphical representation is built using the JavaScript arbor.js[5] library.

Flashing a QR-code inside the Web application: in browsers that support the getUserMedia method of the MediaStreams API, users can take a picture of the QR representing a paper URI and send it to the server. It is then decoded using the ZXing[6] library and the user is redirected to the paper homepage. Of course one can also flash the QR codes using a native application to retrieve the same publication homepage.

3. CUSTOM DATASOURCES

This section focuses on two particular datasources that we designed and developed to be queried by the DataConf mashup.

[5] http://arborjs.org/

[6] http://code.google.com/p/zxing/

3.1 Schedule management

The WWW'2012 experience showed that a conference schedule is a dynamic artifact: session rooms frequently change and paper presentations even take place in different session that the ones initially planned. Printing this schedule necessarily led to an obsolete version of this artifact. In order to provide an accurate version of the time schedule, we designed a separate Web application to facilitate conference schedule management, named **SimpleSchedule**. SimpleSchedule provides conference managers with a convenient means to keep this information up to date. It facilitates the conference schedule management task by providing a calendar-like tool that allows users to graphically handle their events hierarchy. SimpleSchedule is provided as an independent backoffice tool available at https://github.com/fio-ben-TER2013/WWWConference. Its interface is designed as follows:

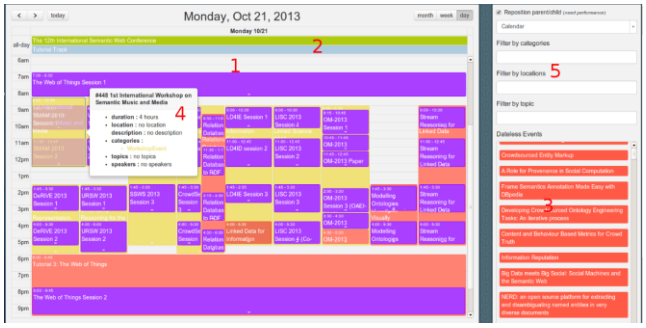

Figure 3. SimpleSchedule backoffice interface.

1. **Central panel**: This panel displays all the conference events included in the chosen timeline. As the SWC ontology proposes a hierarchy of event types (e.g. swc:TrackEvent, swc:TalkEvent, swc:SessionEvent), SimpleSchedule provides a way to easily imbricate events by dragging one event on another. Also, any event can be deleted, replaced or modified directly by simple click on the event box.

2. **All-day event zone**: Some events happen to last a whole day. In this case they are displayed in this box without polluting the central panel.

3. **Dateless event zone**: When building or importing a conference schedule, it happens that some events are uncertain, either because they do not have a specified start date in the imported dataset or simply because they have not yet been scheduled. To facilitate the chair work, the dateless event zone act as an event store where undecided events can be dragged or pulled anytime.

4. **Details pop-over**: In order to avoid central panel overloading, a pop-over is displayed when flying over an event. This pop-over indicates the categories, the acronym, the location, the topics, the presenters of the concerned event.

5. **Filter zone**: As hundreds of events can be contained in the dataset, it is mandatory for the user to have the possibility to apply some filters. The filter zone provides three independent filters: topic, location and category.

On its front end, SimpleSchedule provides a Web service that exposes the conference events in the ICal vocabulary [6], as defined in the Semantic Web Conference ontology format. DataConf requests SimpleSchedule as a datasource, allowing attendees to view the time and location of the different conference events (tracks, sessions and paper presentations) during the conference.

The main schedule view provides a day-per-day overview of the conference timeline, as shown in Figure 4a. From there, users can browse the event tree, access event descriptions, as shown in Figure 4b, add ICS event descriptions to their own calendars and access the publications related to an event.

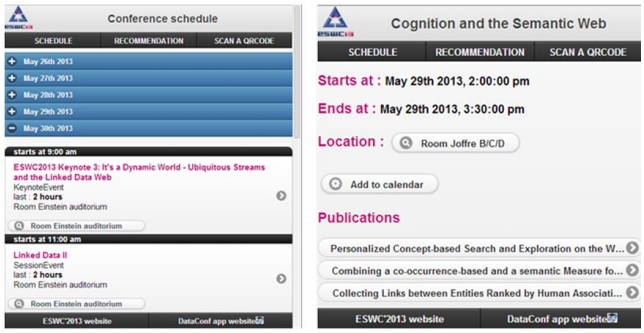

Figure 4. Conference schedule views: a) conference timeline, b) view of a particular session.

3.2 User-enriched data

DataPaper is a custom datasource that allows conference actors (authors, chairs) to provide external information about their own descriptions and publications. This information can be provided using typed links towards external resources, in order to cope with the Linked Data principle. This additional information is intended to be queried by DataConf to enrich the corresponding persons and publications views. As DataPaper can query several conference datasets, DataConf users can benefit of these enrichments during future conferences.

For this, DataPaper stores the URIs, description texts and format of these external resources. These data are stored in key-value pairs, using for instance a publication URI in the conference dataset as a key, the desired property as a description and a String or an URI enriching this publication as a value. The application relies on two stable and widely used vocabularies: FOAF [6] for the profile information and Dublin Core [5] for the publications. As ontologies are evolving quickly and considering that many properties are only at a "Testing" status, these two ontologies are dynamically fetched using Linked Open Vocabulary API[7].

Technically, DataPaper is built on a NoSQL CouchDB[8] database, since it is aimed at retrieving key-value pairs. The account and content management system is powered by WordPress and the business aspects of the application (managing the conference datasets and handling links to external resources) are performed using a non-official plugin (*i.e.* not released on the WordPress platform). The interface and DataConf then queries the service using the URI of an entity present in the conference dataset, in order to retrieve all the external resources that DataPaper stores about this entity.

For instance, when a user adds his/her photo and contact information to his/her account, it will appear on his/her DataConf homepage (*cf.* Figure 2b), regardless of the considered conference.

DataPaper can be accessed through a publicly available Web interface. So that conference chairs do not need to manipulate the DataPaper interface for other reasons than to enrich the conference material, users can automatically create their accounts using the same email they are referenced with in one of the known conference datasets. Hence, conference chairs who have already published their dataset and want their attendees to be able to enrich its content only need to inform the DataPaper admins of the existence of this dataset.

Once logged in, a user is informed of the different resources he/she is responsible, such as his/her profile and publications eventually coming from datasets of different conferences. He/she can enrich each of these resources by adding new links to any of them. Figure 5 shows an example of enriching a publication using the DataPaper interface.

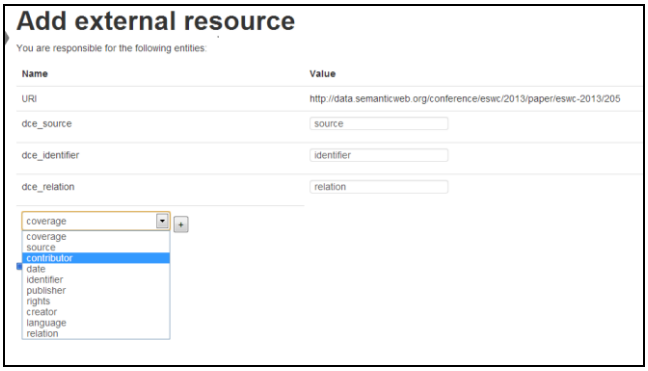

Figure 5. DataPaper publication enrichment interface.

DataPaper allows adding links to resources in a quite generic way. Therefore, its interface still needs to be improved. For enriching one of his/her publications with another external resource, a user specifies the type and URI of the target resource, as well as the type of relation that links both resources together. For this, he/she is assisted by a drop box containing the possible values in the appropriate vocabularies (here, Dublin Core).

These URIs are stored in the database, with the URI of the publication as a key. Each link to an external resource is thus a lightweight element in the database. DataConf can then query DataPaper using this key and decide, using the types of the relationship and of the target resource, of the best means to render this link. For instance, if the resource is an image, DataConf will try to retrieve it from its location and display it to the user, whereas if it is a contact information, it will create an appropriate link without trying to retrieve the resource.

The DataPaper login interface is accessible at http://dataconf.liris.cnrs.fr/datapaper/. Even if it is freely available, we suggest keeping a single DataPaper instance for later conferences, in order to keep the benefits of the contents added by former conference actors about their profiles.

4. CLIENT-SIDE REASONING

Our application also allows browsing publications by keywords, as shown in Figure 2d and getting publication recommendation based on the keywords of the previously explored ones[9]. Both

[7] http://lov.okfn.org/dataset/lov/

[8] http://docs.couchdb.org/en/latest/

[9] As DBLP does not provide information about publications keywords, browsing keywords and publication recommendation are

tasks rely on a keyword ontology that is locally handled inside the mashup application. This requires to first build a keyword ontology, then to load it a locally embedded reasoner and finally to query and exploit it like another datasource.

4.1 Building a conference keyword ontology

Each conference encompasses papers with specific keywords. In order to be able to perform reasoning on the publication keywords of a particular conference, we need to build a keyword ontology specific to this conference. Building such ontologies cannot be done using text-based ontology construction tools, since we only dispose of the publications metadata and not of their full texts. Therefore, we adopted the following methodology. We use the 2012 ACM Computing Classification System [9] (in SKOS /XML)[10] as a basis structure for this ontology. Each of its concepts is extracted and stored in a map M_{ACM}. We gather all the keywords of the dataset publications and store them in map M_{kwd}. We then perform the following process:

Tokenization of the element values of both maps: this is done using the most frequent word separators, such as space, hyphen, parenthesis, etc. The keywords that contain stop words (e.g. "a", "the" or "to") are modified to remove these stop words in M_{kwd}. For instance, "web of data" will be replaced with "web data".

Lemmatization of the map elements: this step consists in retrieving the lemma (root) of each word, to get rid of its flexed form. For this, we use a Java implementation of the Porter Algorithm [10]. In the previous example, "linked open data" will be transformed into "link open data". All elements that belong to the same lemma will be linked with an owl:sameAs relationship.

Structuration: this step aims at building small tree structures of a subset of the keyword map. For each element of M_{kwd}, our algorithm searches for all inclusions of an element in another one and creates an owl:SubClassOf relationship with as domain the current element and as range the included one. For instance, if "link open data" and "link data" are present in M_{kwd}, "link open data" will be considered a subclass of "linked data".

Structure mapping: each element of M_{kwd} that appears as a root element in one of the tree structures defined in the previous step, as well as each element that is an actual keyword and does not belong to one of those structures is re-processed using the same algorithm against the elements of M_{ACM}, to be inserted as a subclass of one of its elements. At the end of this step, lots of the keywords are "hooked" to the ACM taxonomy using parts of their decompositions. The elements of M_{kwd} that were not able to be linked to the M_{ACM} are kept in separate taxonomies. Therefore, we will for the moment keep the structure in our example above that starts with "link data".

Cleaning: the M_{ACM} with the M_{kwd} keywords hooks is processed in order to remove the branches that do not end with leaves that represent actual keywords of the dataset.

4.2 Accessing the keyword ontology

DataConf uses a locally embedded reasoner that processes the on purpose built keyword ontology described above. For this, it uses a modified version of the OWLReasoner [11] JS inference engine. We chose OWLReasoner as it proposes an OWL-EL reasoner and a SPARQL engine and is the most recent freely available JS reasoner. However, it has some limitations: the engine does not reason on individuals nor handle literals, and we had to fix a few issues for being able to send queries using the owl:SubClassOf predicate.

OWLReasoner is intended to be used in three steps: loading the ontology into the engine (in OWL/XML format), launching the classification step (where the engine builds the internal JS objects containing the ontology data) and querying the ontology in SPARQL (using query rewriting techniques to query these internal objects). As these internal JS object mostly consist in an embedded database queriable in SQL, the third step is quite straightforward and could be kept "as is" in our version of the reasoner. However, while performing our first tests, we experienced the following issues during the two preceding steps: 1) we intended to use the reasoner in a Web worker, so that it can be run in background and not block the browser interface. Unfortunately, to preserve loose coupling with the page contents, the code executed in a worker cannot access the DOM, and therefore the window.XMLParser object. It then cannot parse the XML ontology file. 2) Even with a reasoner running in foreground, classifying the WWW'2012 ontology (which contains tens of lines) on a Motorola Dual Core 1Ghz cellphone in Firefox Nightly took more than 1 hour ½.

We then pre-process the ontology on a more powerful machine and export the minimal necessary resulting objects of the classification process in JSON, so that they can be loaded by the reasoner engine, inside the Web worker. The engine can then perform reasoning (since the ontology can still be enriched locally) and respond to SPARQL queries in real time. As the ontology is loaded in background and is cached after its reception, the ontology size has mostly no effect on the application performances.

In order to allow the router to query the reasoner embedded inside the browser, we defined a basic query-response protocol on top of the message-oriented asynchronous communication scheme imposed by the Web worker specification. The router then sends SPARQL queries to the reasoner and the reasoner answers in JSON. Therefore, the exchanged messages are requests and responses, similar to those present in the HTTP transactions. Thus, its data can be processed and integrated in the views in the same manner as if it was a distant datasource.

4.3 Using the ontology

Dataconf thus embeds a custom "local datasource" that performs mobile client-side reasoning on the conference publication keywords. During user navigation, DataConf stores the keywords of the publications viewed by the user. The reasoner allows retrieving super and sub-keywords (and thus the publications they refer to), and recommending to users publications they did not yet see (e.g. referring to keywords close to those they have already explored). Figure 6 shows an example of recommendation interface in the DataConf ESWC'2013 instance. The publications in grey have already been viewed by the user and the ones in blue are those recommended by the system.

only accessible for publications that have an entry in the conference dataset.

[10] Available at http://www.acm.org/about/class/class/2012

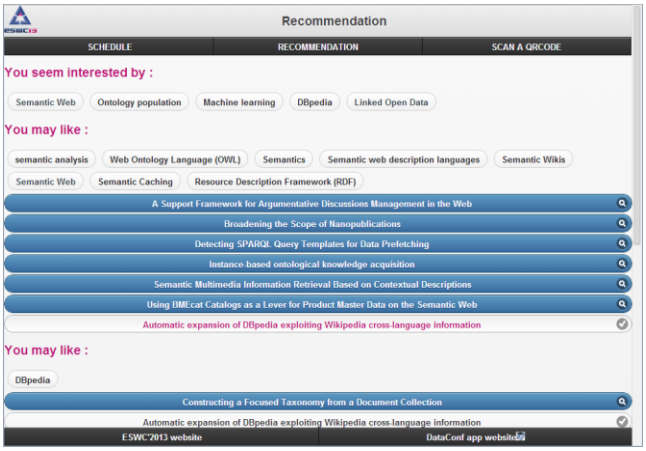

Figure 6. Example of Dataconf recommendation results.

5. ARCHITECTURE

DataConf is a full client-side, single-page Web mashup application. It is built on the widely used Backbone.js[11] JavaScript framework. Backbone provides a routing system that listens to the URL ("hash") change event and maps the current URL to a route that will trigger a specific callback function[12]. DataConf also relies on: 1) the RequireJS library that allows dependency management and lazy loading of the application modules, 2) the jStorage.js library, that provides a cross-browser API for accessing the LocalStorage object, 3) the jQuery Mobile UI toolkit that provides a powerful cross-device widget management system, 4) the history handler embedded in Backbone that provides a ready-to-go page navigation support.

In order to cope with datasources heterogeneity problems, DataConf relies on an internal representation format that can differ from the data format contained in the response. Using this format, data coming from a given source are stored in the browser local storage area before being displayed in the views. This process aims at: 1) caching these data for performance concerns, 2) waiting for data coming from several sources to be ready before starting the data composition and rendering process 3) enabling service composition by embedding previously retrieved data in future queries.

On top of the Backbone.js framework, we designed the component-based architecture that constitutes our mashup application. The main components of this architecture are a workflow engine that can send queries to and process responses from different datasources and a specific template loading system. All these components are configurable, so that the DataConf architecture can be reused for creating another mashup in a different domain. In this section, we first detail this architecture and explain how to reuse it to create other mashups by adding datasources and create the corresponding processing components.

5.1 Architecture components

The mashup architecture is depicted in Figure 7 and organized around the following components.

Router: the core of the DataConf engine is an on purpose built Backbone.js router that somehow plays the role of application controller. At initialization time, it interprets the JSON configuration file (see next section) and identifies the requested resource types and related commands. When a new hash is selected, it dynamically creates the corresponding route by setting a list of commands associated to datasources and triggers the AjaxLoader for each of these commands. At response time, it triggers the model and view callbacks.

Commands: Each command is a set of JS functions that define how to send, process and render a particular request type to a datasource. For instance, there is a command to retrieve the publications of an author from the DBLP SPARQL endpoint. Commands contain the following fields: name (*i.e.* id), HTTP request method and returned datatype (RDF / JSON), as well as three methods:

1. getQuery: constructs a query object to be sent to the datasource and containing either a SPARQL query (for SPARQL endpoints) or a set of parameters (for Web APIs).
2. modelCallback: handles the response from the datasource, performs calculations over the response data, transforms the results into the appropriate common representation (depending on its nature: title, author name, homepage…) and stores them using the LocalStorageManager component.
3. viewCallback: is triggered by the router to integrate these formatted data into the views.

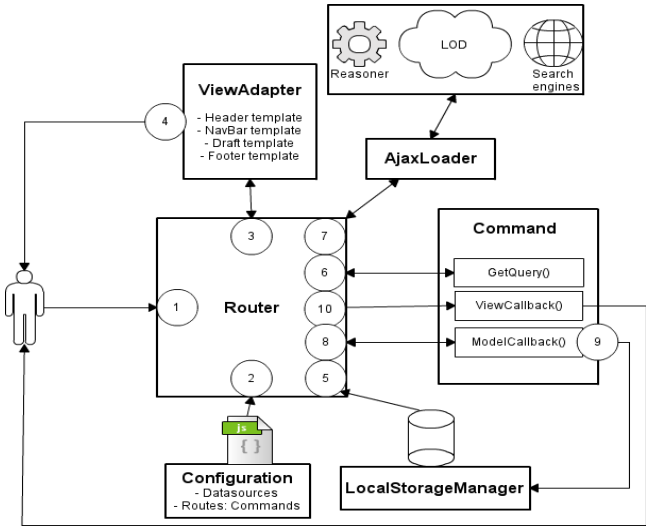

Figure 7. View of the DataConf layer on top of Backbone.js.

AjaxLoader: the AjaxLoader is the component that actually sends the requests. For this, it uses information available in the datasource (URI and cross-domain mode[13]) and in the command (HTTP method, returned type, query along with its parameters and the model callback function).

LocalStorageManager: It is a wrapper to jStorage.js. It handles homogeneous data in the browser LocalStorage area.

[11] http://backbonejs.org/

[12] DataConf URL hashes start with a SWC resource type, followed by several parameters, one of them being the encoded URI of the desired instance.

[13] A client-side mashup application violates by essence the AJAX same-origin policy, since it needs to send cross-domain requests. Two techniques allow sending such requests from a browser: JSONP [12] and CORS [13].

ViewAdapter: This component is a wrapper to jQueryMobile. During application loading, it is set up with the templates needed to build a DataConf view, namely: footer, navbar, header and draft view (i.e. empty content block to be filled with the data to be fetched). At runtime, it performs the view rendering tasks, such as page change management, widget generation and layout preparation.

Configuration: See section 5.3.

5.2 Route processing

Figure 7 enlightens the interactions between the application components. After receiving a hashchange event, the application reacts as a succession of processes, all triggered from the router. The datasource querying and integration process in DataConf is sequenced as follows.

1. The user changes the current page to view a specific resource.

2. The router catches the hashChange event and reads the requested URL. By comparing the resource type with the routes declared in the configuration file, it retrieves the list of commands to execute and the associated datasources.

3. The router triggers the rendering of the new page by calling the ViewAdapter.[14]

4. The ViewAdapter triggers the page change event, prepares the new layout and pushes the prepared layout to the page DOM.

5. For each command defining a request to a datasource, the router checks if the result is available in the local storage area. If it is, the router bypasses the datasource querying process and goes directly to step 10.

6. For each command, the router prepares a request by calling the getQuery() method. This method returns the final query string (including the URI of the resource to view and its other parameters), as well as the expected response MIME type.

7. The AjaxLoader sends the provided query to the datasource related to the command, using the datasource configuration (endpoint URL, cross domain mode).

8. When receiving Ajax results, the router triggers the command model callback. It eventually performs processing on the results and transforms them into the internal format previously described, before returning them.

9. The model callback pushes the results in the storage.

10. The router calls the view callback to render the results in the prepared layout.

5.3 Configuring a DataConf instance

To be deployed for a particular conference, DataConf relies on a JSON configuration file. For the instance to be relevant, it must at least contain a datasource SPARQL endpoint hosting the conference SWC dataset within its declared datasources. The configuration file encloses the following elements:

Conference general information: name, acronym, homepage, base URI and logo URI

Datasources: endpoint URI, supported cross-domain methods, associated CommandStore (see below)

Routes: URL hash path, corresponding view template id, list of associated command ids.

All the commands associated to a given datasource are stored in a separate JS file called a **CommandStore**. A CommandStore contains a JSON table, each element describing a command, using the following elements: id, response dataType, HTTP method, getQuery, modelCallback, viewCallback.

5.4 Developing a datasource component

Developers who wish to add a new datasource to their DataConf instance are welcome to do so. For this, they need to build the datasource CommandStore by describing all the commands associated to requests that can be sent to this datasource. Once this done, they need to declare this datasource, link it to their CommandStore and associate the commands with the appropriate routes in the configuration file.

6. DISCUSSION

As experience about building DataConf instances grows, we found out that the major difficulty we faced while issuing DataConf instances for a new conference venue was to gain access to a complete and valid SWC ontology describing the targeted conference metadata. Setting up such an ontology is definitely a big challenge for conference organizers. The first reason is the difficulty to gather all conference materials in a common format. Indeed, the data are usually handled by different session chairs who may use different tools and formats. It is not usually difficult to retrieve information about the main research tracks, but it is harder to dispose of a comprehensive dataset, including all other sessions such as posters, demos and even workshops. Moreover, designing an SWC ontology from the data expressed in another format is a highly time-consuming task for semantic Web specialists and a barely unrealizable one for neophytes.

As DataConf takes its main asset from its live usage by conference attendees, the second difficulty is to keep the data exposed by a conference endpoint up-to-date, especially concerning the schedule. Yet, conference schedule changes are not a rare thing. Propagating each change by modifying an RDF file and uploading it on a triple store is something that no conference organizer would be willing to do. By developing the SimpleSchedule application, we tried to minimize the impact of such problems.

However, even using a user-friendly tool such as the SimpleSchedule interface can be considered a waste of time if it is not integrated in the whole conference management process. Hence, our next move will be to propose a comprehensive solution that will take place after the reviewing process (when the accepted papers lists are available) and assist conference organizers before and during the conference. It will allow them to import/export the conference metadata from/into different formats, store them in a regular database and of course easily allow modifying the schedule of the presentations and exposing them in a SPARQL endpoint in the SWC ontology format[15]. Moreover, as DataConf is

[14] Steps 3 and 4 must be performed before sending the requests to the datasources, so that the interface does not seem to freeze while awaiting network communications.

[15] Using for instance a D2RQ [14] solution to map the field of the relational database to the SWC individuals.

lightweight and easily configurable, it will also generate an instance for a particular conference on the fly.

If it happens to be stable and last after the conference venue, this tool will then enrich the LOD cloud and be usable by other semantic mashup applications. We could for instance imagine that DBLP uses a dump of a conference dataset available on this endpoint to add the publications to its database. Bibliographic tools could also query conference endpoints to recommend publications to their users.

Conference metadata represent a great educational value for students, Phd students and researchers. Our point here is to say that the SWC ontology seems far from reaching the success it deserves. We believe that the scientific community could be the first one to benefit of the semantic Web advances and that there is a number of use cases that can benefit of exchanging publication metadata, before, during and after a conference.

Our last point is that issuing endpoints is not sufficient to have them adopted and integrated in semantic tools. Another condition to leverage the use of conference metadata is to link those data to the LOD cloud. This is the purpose of the DataPaper application that connects conference resources to external ones. We willingly restricted the possible link types to two vocabularies (DC and FOAF), in order to limit the variability of the targeted resource types. Indeed, for each resource present in a DataPaper response, DataConf has to decide on the fly how to handle it. This is currently done using a simple switch between the known MIME types. In order not to limit the user's resource choice, we are currently working on using REST HATEOAS [15] and Linked Services [16] techniques to carry, together with the DataPaper response, the appropriate service endpoint to handle the resource.

7. CONCLUSION

In this paper, we have presented DataConf, a mobile Web application that proposes various features among which accessing and browsing conference publication metadata. It has been designed w.r.t. recent Web standards and technologies and so that as much computation as possible is done on client side. DataConf uses cross-domain requests to access and enrich the metadata from different SPARQL endpoints and Web APIs. We also presented the DataConf ecosystem, consisting in several datasources, thanks to which these data can be enriched both by the conference organizers who wish to update the conference schedule, as well as by the other attendees who want to link the conference material to external resources. The application also locally manages a keyword ontology and uses it for recommending publications to the user. DataConf architecture is modular, extensible and can be reused for other types of client-side mashups. Several DataConf instances have been released since its first version for the WWW'2012 conference, allowing us to provide a short feedback about the place of such applications in the scientific conference management process, as well as how to leverage Linked Open Data techniques to help this process.

Based on these reflections, we currently intend to pursue our work in two directions. An operational direction conducts us to develop the SimpleSchedule datasource to a wider solution that helps manage conference organization and exposes conference metadata on a SPARQL endpoint. A more theoretical direction is to improve our mashup architecture to take greater advantage of its embedded inference engine and to enable linked services support.

8. REFERENCES

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

[2] Médini L., Bâcle F., Nguyen H.D.T.: DataConf: enriching conference publications with a mobile mashup application. In WWW (Companion Volume) 2013: 477-478

[3] Möller, K., Heath, T., Handschuh, S., Domingue, J.: Recipes for SemanticWeb dog food - The ESWC and ISWC metadata projects. In: 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC2007+ASWC2007), 11-15 Nov 2007, Busan, South Korea (2007)

[4] Möller, K., Bechhofer, S., Heath, T.: Semantic Web Conference Ontology, http://data.semanticweb.org/ns/swc/ontology (2009)

[5] Dublin Core Metadata Initiative. "Dublin core metadata element set, version 1.1." (2008).

[6] Brickley, D., & Miller, L. FOAF vocabulary specification 0.98. Namespace document, 9. (2010).

[7] Internet Calendaring and Scheduling Core Object Specification (iCalendar) - RFC 2445 - IETF. November 1998, http://www.faqs.org/rfcs/rfc2445.html (1998)

[8] Ley M., The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives. In Proceedings of SPIRE 2002, String Processing and Information Retrieval, 9th International Symposium, Lisbon, Portugal, September 11-13, 2002. Springer, Lecture Notes in Computer Science, Alberto H. F. Laender and Arlindo L. Oliveira Eds., Vol. 2476, pp. 1-10. ISBN: 3-540-44158-1 (2002).

[9] Lillian N. Cassel, Sudhamsha Palivela, Srikanth Marepalli, AhiMahidhara Padyala, Rahul Deep, and Siddhartha Terala. 2013. The new ACM CCS and a computing ontology. In Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries (JCDL '13). ACM, New York, NY, USA, 427-428. DOI=10.1145/2467696.2467780 http://doi.acm.org/10.1145/2467696.2467780

[10] Karen Sparck Jones and Peter Willet, 1997, Readings in Information Retrieval, San Francisco: Morgan Kaufmann, ISBN 1-55860-454-4.

[11] OWLReasoner JavaScript Inference engine. Available at: http://code.google.com/p/owlreasoner/

[12] Ippolito, B., 2005. Remote JSON – JSONP. Available at: http://bob.ippoli.to/archives/2005/12/05/remote-json-jsonp/

[13] Cross-Origin Resource Sharing. W3C Candidate Recommendation, 29 January 2013 : http://www.w3.org/TR/cors/

[14] Bizer, C., Seaborne, A., D2RQ-treating non-RDF databases as virtual RDF graphs. In Proceedings of the 3rd International Semantic Web Conference (ISWC2004), p. 26 (2004)

[15] Fielding, R. T. (2000). Architectural styles and the design of network-based software architectures (Doctoral dissertation, University of California).

[16] Pedrinaci, C., & Domingue, J. Toward the next wave of services: linked services for the web of data. Journal of Universal Computer Science, 16(13), 1694-1719. (2010)