# A Drag-and-block Approach for Linked Open Data Exploration

Tuan-Dat Trinh, Ba-Lam Do, Peter Wetz, Amin Anjomshoaa,
Elmar Kiesling, and A Min Tjoa

Vienna University of Technology, Vienna, Austria
{tuan.trinh,peter.wetz,ba.do,amin.anjomshoaa,
elmar.kiesling,a.tjoa}@tuwien.ac.at

**Abstract.** Since its initial definition in 2007, the concept of Linked Open Data (LOD) has gained strong traction in the scientific community. However, mainstream adoption has been limited and the emergence of an envisioned global linked data space is still in its early stages. One possible explanation is the gap between the large amounts of published LOD datasets and the lack of end-user tools to effectively explore them. Because existing applications are tailored towards specific datasets and do not allow for reuse and extension, novice users have so far had limited means to access the rich data sources being published. To address this issue, we introduce a novel approach to support non-expert users in the flexible exploration of LOD. To this end, we define a formal model that makes use of existing links between interconnected datasets. We implement the model in a mashup platform and illustrate its potential by means of use cases combining Open Data and Linked Open Data sources.

## 1 Introduction

Linked Data (LD) refers to "*a set of best practices for publishing and connecting structured data on the Web so that it can be interlinked and become more useful*" [2]. Built upon standard web technologies, e.g., HTTP, RDF and URIs, it has been adopted and applied in the LOD project – a community project supported by the W3C – to facilitate the publication of open datasets as RDF. At the very beginning, there were only twelve published datasets, but the so-called LOD cloud grew rapidly to 928 datasets with 62 billion triples by 2014[1]. Although these data sources provide highly valuable data with enormous potential for interesting applications, the LOD cloud today is still largely a collection of raw datasets and making effective use of them remains a major challenge.

Although researchers and practitioners have recently implemented many applications based on LOD, few of them specifically target end users with limited or no experience with semantic web technologies. Therefore, gathering data from multiple LOD resources and performing data analysis, integration, and visualization tasks still remains a cumbersome process. Furthermore, most existing

[1] http://stats.lod2.eu

applications – with the exception of dedicated query and data integration tools – make use of only a single or a limited number of particular LOD datasets. As a consequence, LOD is almost exclusively processed by custom applications tailored to specific use cases or domains and remains inaccessible to end users with more general information needs. This situation is not consistent with the original vision of the Semantic Web, which promised to facilitate easy discovery, sharing, and reuse of highly interconnected datasets across applications.

In this paper, we first address the challenge to support end users in obtaining, integrating, and visualizing data from different LOD datasets. We also address the question how end users can benefit quickly from new data sources added to the LOD cloud.

Following the Linked Data principles [2], each published LOD dataset should include links to external datasets, e.g., DBpedia [14]. To researchers, this large collection of linked resources is useful for data integration, but to end users, those links are currently solely used for resource navigation. Therefore, secondly, we address the challenge of leveraging the interconnected links between datasets and thereby contribute towards fulfilling the primary objective of Linked Data.

In Section 2, we introduce our model as an approach to overcome these two challenges. Generally, end users have a wide range of varying requirements with respect to heterogeneous data. Because it is impossible to create a custom application for each individual user requirement, our approach is to modularize the required functionalities into blocks that users can recombine arbitrarily to create new applications. Conceptually, the model represents a consistent framework for creating, managing and synthesizing reusable LOD-consuming applications.

More specifically, data publishers and Semantic Web developers first define scenarios to consume and combine their LOD dataset with others. Next, the tasks of collecting, processing, integrating, and visualizing data are encapsulated into *Linked Blocks*. Each *block* represents a miniature application working with a specific part of a LOD dataset. Finally, end users select and combine *blocks* to dynamically compose LOD-consuming applications, which later can be shared or published on the web.

A key concept is that *blocks* can be connected to each other. By simply connecting *blocks*, end users can leverage the *linked* nature of the LOD cloud without an understanding of the intricate details of queries and transformations executed behind the scenes. Developers can create new *blocks* easily and users can reuse these blocks in multiple scenarios. The potential for interesting applications consuming different LOD datasets is hence limited only by the creativity of end users.

The remainder of this paper is organized as follows. In Section 2, we present our LOD block model and introduce the *block* concept; in Section 3, we introduce a mashup platform based on the proposed model. Section 4 illustrates the potential of the approach by means of example use cases. We provides pointers to related work in Section 5 and describe our preliminary evaluation in Section 6. The paper concludes in Section 7 with an outlook on future research.

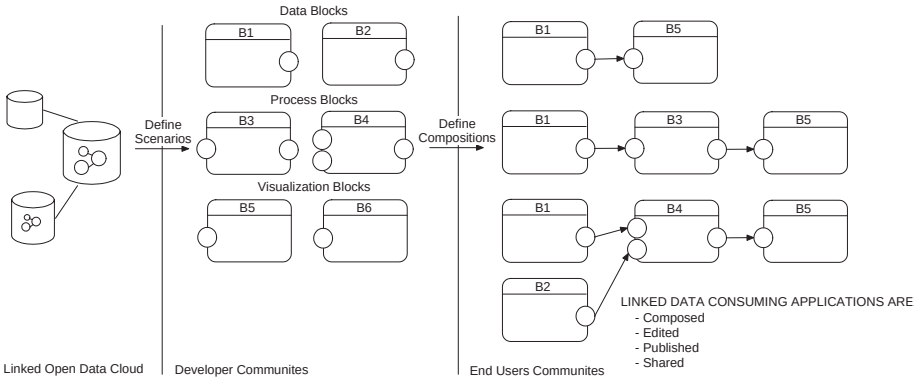



Fig. 1: LOD block value chain

## 2 LOD block model

The idea of our LOD block value chain illustrated in Fig. 1 combines *Web Services* and *Service-Oriented Architecture (SOA)*[15] concepts. Whereas services target developers, we transform them into *blocks* aimed at end users. To utilize data from different LOD datasets, publishers and developers can create three types of *blocks*: (i) *data blocks*, which collect data from one or multiple datasets; (ii) *process blocks*, which process and combine data in different ways through enrichment, transformation, and aggregation; and (iii) *visualization blocks*, which display the final data. These blocks are organized into three layers, i.e., *data layer*, *business layer*, and *presentation layer*.

These blocks can be used by end users to compose LOD-consuming applications in many ways and create a value chain between LOD, developers and end users. This model enhances the reusability of the developers' work; it stimulates end users' creativity to build up dynamic applications; and it inherits various benefits of *SOA* and *Software Component-Based* approaches.

*Blocks* allow non-expert users to build ad-hoc applications rapidly. Each block can receive input from other blocks to process and return its output which, in turn, can serve as input for another block. To end users, blocks are "black boxes" with adjustable parameters that control the *processing function*. Similar to functional programming or web services, blocks can have multiple input gates, but only a single output gate.

The input/output data model is an RDF graph representing a list of instances of an RDF class. Each instance is addressable via a URI and annotated with data and object properties. The value of an object property, in turn, can be another instance. Such instances and the initial instances of both input and output can have mutual relations. Fig. 2 shows input, output models of a block which takes Cities as input and return all Laureates born there; a *City* from input is linked with the corresponding *Person* from output via the *birthPlace* relation.

Two mandatory components for a complete application are *data blocks* and *visualization blocks*. A *data block* does not have any inputs; it collects data from

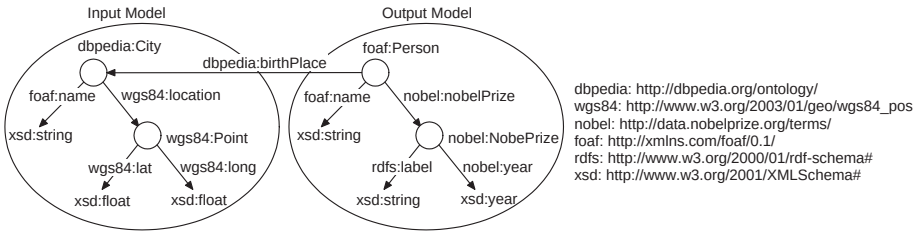



Fig. 2: Example input and output models of a block

an arbitrary data source to provide input for *process* and *visualization blocks*. *Visualization blocks* display output data of *process* or *data blocks*. Inside an application, more than one *visualization blocks* can be applied, because there can be multiple ways to display the same data.

To compose an application, users simply select appropriate blocks and connect the output of a block to the input of another one. The model checks whether the semantics of the particular input and output data models match and only allows for connections between compatible gates. There are four types of links: (i) links between *data blocks* and *process blocks*, (ii) links between *data blocks* and *visualization blocks*, (iii) links between two *process blocks*, and (iv) links between *process blocks* and *visualization blocks*.

Corresponding to the three possible operations of *process blocks*, i.e., enrichment, transformation, and aggregation, there are three sub-types. If blocks get additional data from other datasets and add it to the input, they are called *enrichment process blocks*. An example is a block in which its input – a list of *Locations*, i.e. $[Locations(lat, long, description)]$ – is enriched by sample images for each *Location*, i.e. $[Locations(lat, long, description, [Image])]$.

*Transformation process blocks* transform input instances into output instances of a different RDF class based on their interrelations. Hence those blocks are the crucial element which enables users to leverage the key ideas of LOD, i.e., making use of links between resources from one or more LOD datasets. For example, a block may receive *Locations* and output $MusicEvents$ organized at a place nearby. The final type, *aggregation process blocks*, aggregates data from its multiple input gates. Therefore, blocks of this type always have at least two input gates representing different entities of similar classes.

Formally, each block is represented by a quadruple $(I, R, O, C)$ with a finite set of graph-based input models $I = i_1, \ldots, i_n$, a graph-based output model $O$, a configuration model $C$ and a processing function $R : i_1 \times \ldots \times i_n \times C \rightarrow O$. Whereas *process blocks* require all elements (i.e., $I, R, O, C \neq \emptyset$), *data blocks* have no input (i.e., $I = \emptyset$) and *visualization blocks* have no output (i.e., $R, O = \emptyset$).

To sum up, major advantages of the block concept are (i) the flexibility in building applications, starting from selecting data sources, applying various processing operations, to visualizing the final data in multiple ways; and (ii) the effective use of links, i.e., inner links and outer links, between LOD datasets.

## 3 Linked Widgets Platform

We define a number of design objectives before implementing the model and an initial set of blocks. First, blocks need to run on heterogeneous platforms and communicate with each other; the cost to develop blocks should be low; everyone should be able to contribute blocks; and it should be easy to share complete applications composed from disparate blocks. Using modern web technologies, we implement each block as a widget, i.e., "*an interactive single purpose application for displaying and/or updating local data or data on the Web, packaged in a way to allow a single download and installation on a user's machine or mobile device*"[2].

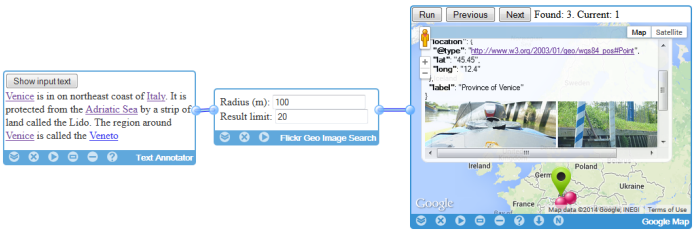
Our mashup platform, a web platform consuming open data sources to make them accessible and processable for end users, is an implementation of the model presented in Section 2. More detailed information on implementation aspects can be found in [10,11]. Using arbitrary web languages, developers can create widgets and deploy them on arbitrary servers. Currently, both widgets and mashup metadata are being published according to the first three of the four LOD principles. To accommodate the fourth LOD principle, we also aim to establish connections with other LOD datasets in the future. Widgets are RDF instances whose URIs can be de-referenced for detailed information. Their input and/or output models are semantically annotated and are accessible through a SPARQL endpoint. Hence, the platform operators or third parties can develop widget-matching and widget-combining algorithms. Widgets are grouped into widget collections used by end users to compose mashups or ad-hoc applications. These mashups can be created, edited, auto-parameterized, stored, shared, executed, displayed, and, in turn, reused as new widgets.

## 4 Example Use Cases

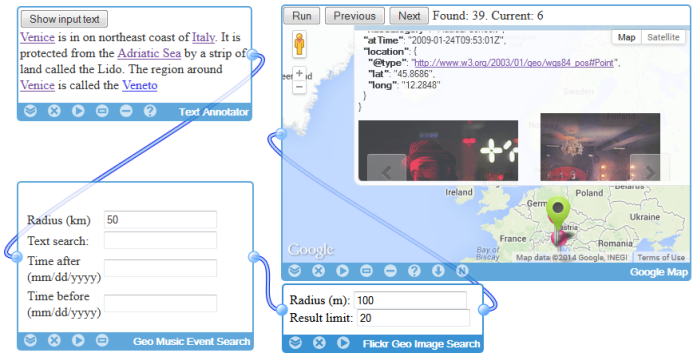
Based on geographic objects with latitude and longitude properties, datasets in the Linked Open Data cloud can be queried dynamically and linked to each other. We use five LOD datasets and two other open data sources in our examples:

1. `http://linkedgeodata.org` – Publishes data collected by the Open Street Map project as RDF data.
2. `http://www.geonames.org` – Contains more than 10 million geographical names and consists of over 8 million unique features including 2.8 million populated places and 5.5 million alternate names. *GeoNames* data are linked to DBpedia and other RDF datasets.
3. `http://spotlight.dbpedia.org` – A service looking for approximately 3.5 million things of unknown or 320 known types in text and linking them to their global unique identifiers in DBpedia.
4. `http://eventmedia.eurecom.fr` – A LOD dataset composed of events and media descriptions associated with these events which are obtained from three large public event directories, i.e., last.fm, eventful and upcoming.

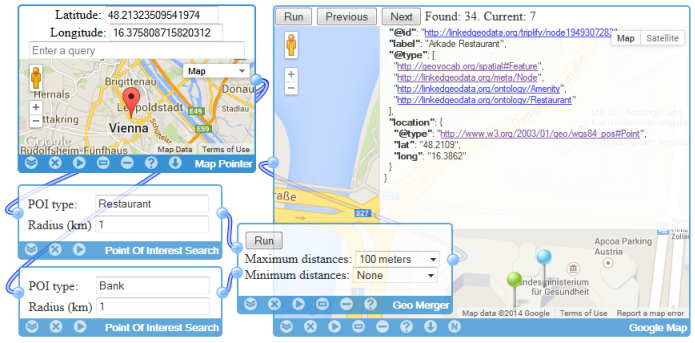
[2] `http://www.w3.org/TR/widgets-reqs/`



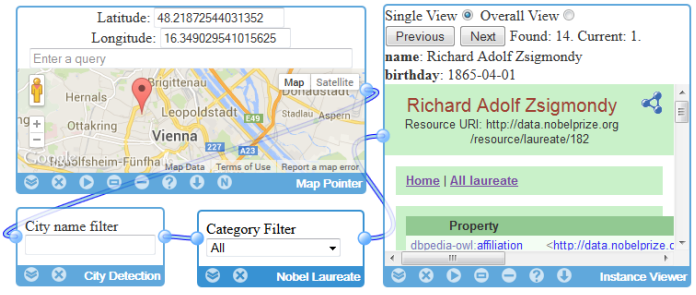

(a) Display detected-from-text famous Places (and Images) on the map




(b) Display Music Events (and Images) nearby detected famous Places on the map




(c) Nearby Restaurants and Banks that are less than 100 meters from each other




(d) Nobel Laureates born in a City

Fig. 3: Example use cases

5. `http://data.nobelprize.org/snorql` – Contains information about who has been awarded the *Nobel Prize*, when, in what prize category and the motivation, as well as basic information about the laureates.
6. `https://flickr.com` – An image and video hosting website. It provides a free API to access 5 billion photos with valuable metadata such as tags, geolocation, etc.
7. `http://map.google.com` – Offers satellite imagery, street maps, and street view perspectives which can be accessed through its API services.

### 4.1 Block Annotation

In our example use cases, we use ten blocks organized into three layers, i.e., data, business, and presentation layers.

The two *data blocks* are *Text Annotator* (1) which detects a list of Places from text, and *Map Pointer* (2) which outputs one or multiple points chosen by end users on a map.

In the business layer, we have six *process* blocks. *POI Search* (3) receives any kind of objects with lat and long attributes and transforms them into Points of Interest (POIs) located nearby. Similarly, *Music Event Search* (4) returns music events. *City Detection* (5) uses the GeoNames service to find Cities that the input objects belong to and looks up extra information via DBpedia, e.g., area or population. *Nobel Laureate* (6) takes Cities as input and returns a list of Laureates born in those Cities. *Image Search* (7) enriches all types of geographic objects with Flickr images. *Geo Merger* (8) aggregates two lists of geographic input objects based on the distances between them.

Finally, there are two blocks in the presentation layer: *Google Map* (9) displays the input geographic objects one by one on a map, along with their properties. *Instance Viewer* (10) shows input objects one by one and the objects' corresponding URI-dereferenced page.

For each block, out of the quadruple $(I, R, O, C)$, only two components, i.e., $I$ and $O$, are semantically annotated. It is unnecessary to annotate $R$ and $C$, because they are encapsulated inside a single block and are not used in the block-combining process (i.e., $R$ and $C$ of one block have no relation with the $R$ and $C$ of any other block). Detailed information about the blocks and their annotated input and output models can be found on our mashup platform at `http://linkedwidgets.org`.

### 4.2 Example Block Combinations

There are many ways to compose useful applications from the ten blocks selected for our examples. Assume, e.g., that we have a touristic text introducing different beautiful spots in a city. The application depicted in Fig. 3a then presents on overview for those places. Next to detailed information from DBpedia, the locations and images are displayed on the map. Then, if we add the *Music Event Search* between the *Text Annotator* and the *Image Search*, a different application finding music events near detected famous places is created as shown in Fig. 3b.

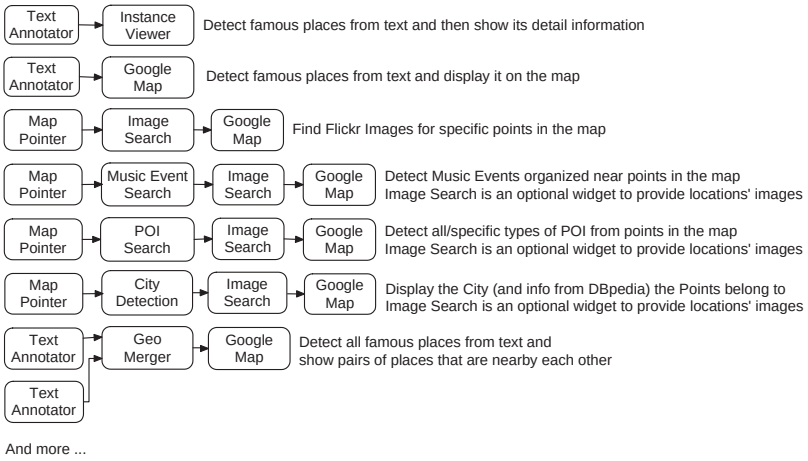



Fig. 4: Example block combinations

From a point specified by a user on the map, the combination in Fig. 3c finds all pairs of nearby restaurants and banks that are less than 100 meters away from each other. We illustrate another use case in Fig. 3d, listing all Nobel laureates born in the city based on a map input point.

As shown in Fig. 4, there are more ways to combine the example blocks. We can create new applications by replacing two *data/visualization blocks* with each other. Moreover, one or more *process blocks* can be added between a *data* and a *visualization block*. By linking the blocks, the data from different LOD datasets are connected. Furthermore, with different values of $C$ inside a block, new use cases can be created. In Fig. 3c, we can replace "Bank" by "Park", and add one more instance of both *POI Search* and *Geo Merger* to find combinations of restaurant, park and book shop near each other.

## 5 Related Work

Multiple mashup tools using a widget/block-based approach have been developed, e.g., JackBe Presto Wires[3], Microsoft Popfly[4], Yahoo Pipes[5], Openkapow[6], Lotus Mashups[7], DERI Pipes [1], MashQL[21], Super Stream Collider [20] (SSC), ResEval Mash [17], and Dashmash[19]. However, some of them have been discontinued, many of them are domain specific, and their blocks or operators are typically rather limited. With the exception of DERI Pipes, MashQL, and SSC, they do not aim at handling semantic data.

[3] `http://mdc.jackbe.com/prestodocs`
[4] `http://en.wikipedia.org/wiki/Microsoft_Popfly`
[5] `http://pipes.yahoo.com/pipes`
[6] `http://kapowsoftware.com/products/whats-new-in-9.3/index.php`
[7] `http://www-10.lotus.com/ldd/mashupswiki.nsf`

Whereas SSC is a mashup tool for live stream data, MashQL is a generic tool which allows users to easily create a SPARQL query, using its own query-by-diagram language. MashQL cannot aggregate data from different sources and its output visualization is restricted to text and table formats.

DERI Pipes enables users to perform semantic data processing tasks from different RDF data sources. Its input can be RDF, SPARQL query results, XML, or HTML. Therefore, potential users need knowledge of these formats and how to process them algorithmically to make effective use of the provided operators.

Similar limitations apply to a Linked Data Integration Framework presented in [5], a semantics-enabled mashup of existing Web APIs [13], and a web-based method that integrates static and dynamic sources for Linked-Data consuming applications [8]. In other words, those are data integration frameworks aimed at developers, not end users. They can be encapsulated in the *process blocks* of our model to facilitate new usage scenarios.

The contributions [9] and [12] discuss useful semantic mashup systems, e.g., [16], [18], but lack a systematic approach for all LOD datasets. Although each system can effectively exploit only one or several datasets, they cannot be extended to more LOD datasets and do not utilize LOD interconnections. Yokohama Art Spot [7], for example, is a web mashup application that offers information on art in Yokohama by consuming three LOD dataset (LODAC Museum, Yokohama Art LOD, and PinQA) and hence a domain-specific solution.

In general, there has so far been limited research on allowing end users to consume data in a dynamic way. Typically, users are expected to use semantic browsers to explore data and collect information by themselves. Alahmari et al. [6] evaluate fourteen semantic browsers (such as Sigma, Marbles, Disco, etc.) with respect to consumption of structured Linked Data. Those browsers do not offer means to combine data from different sources.

## 6 Preliminary Evaluation

To evaluate our Linked Widgets (LW) platform from a user perspective, we conducted an experiment with two subjects familiar with Semantic Web concepts and solid web programming skills. We asked the subjects to implement a simple use case using our LW platform as well as DERI Pipes, and contrasted the two results. We chose the latter because despite significant differences, it appears to be the most comparable tool in terms of goals and scope. We explained the idea and basic functionalities of both tools to the subjects.

As an example, we asked subjects to detect places from a text and fetch latitude, longitude, and DBpedia URIs for those places. The subjects as a pair spent 2.5 hours to create the mashup illustrated in Fig. 5 in DERI Pipes. The six steps involved in the process were: (i) using a *URL builder* operator to call the DBpedia annotator service, (ii) converting the HTML result into XML format, (iii) using an online XSLT transformation to convert the XML result into RDF format, (iv) executing a SPARQL query over the RDF result to extract URIs of detected locations, (v) for each URI, fetching RDF data from the corresponding

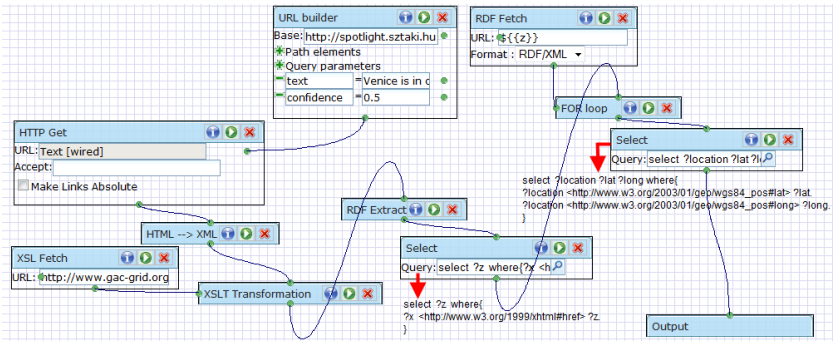



Fig. 5: Evaluation use case implemented with DERI Pipes

deferenceable DBpedia page, (vi) executing another SPARQL query to get the latitude and longitude of each place. The final output of this mashup is served as raw RDF and DERI Pipes does not provide any means to display it visually on a map.

During the experiment, the two subjects found it difficult to apply operators, because it was unclear to them which operators could be connected with others. Thus, they had to find out via trial and error. Because DERI Pipes cannot process non-semantic data, subjects had to convert HTML results into RDF format and execute a query to extract the desired URIs. Moreover, enriching extracted location URIs with Flickr images is not possible in DERI Pipes. After the experiment, subjects stated that for this use case, they would prefer to directly program in code rather than using DERI pipes as a visual programming tool; a direct coding approach, in their opinion, seems more straightforward. For instance, the for loop in DERI Pipes is restricted to semantic data only and therefore not usable in a general way.

Using the LW platform, subjects first used the keyword-based and semantic widget model-based search to find *Text Annotator* as their needed widget. They then ran the widget and saw the result data, and the experiment was completed within minutes. Moreover, they can use the terminal matching feature from *Text Annotator*'s output terminal to identify which widgets can be connected with this output terminal. This feature will instruct them to display the final result in the *Google Map* widget, which is not possible in DERI Pipes.

From our experiment, we found that the most apparent difference between the LW platform and DERI Pipes from a user perspective is: DERI pipes is low-level data processing oriented whereas LW platform acts on a higher level and is more problem-oriented. Users working with DERI Pipes have to be familiar with special technological concepts, e.g., URL, XSLT transformation, XQuery, and hence face considerable difficulties. In using the latter, users first define a goal, e.g., search for POIs near a place, then can discover appropriate widgets, and finally arrange them in a mashup. To ease this process, we organize widgets in a taxonomy tree and in domain-specific collections and provide keyword and semantic search features based on the widget model.

## 7 Conclusion and Future Work

This paper presents an approach for integrating multiple LOD datasets by leveraging their interconnections in a systematic and scalable manner. The key idea introduced is to modularize functionalities into blocks, which can be combined in order to enable users to dynamically obtain, enrich, transform, aggregate or visualize data in different ways. Because these blocks are developed and used in an open manner, the more creative developers and end users are, the more powerful our block model can become. To create a smart data exploration environment, we implemented the model in an open mashup platform.

Unlike similar approaches, the LW platform focuses on high-level data processing and acts as a problem-oriented mashup system. Its widgets are backed by a semantic model to facilitate input-output model matching and widget auto-composition features. Widgets can obtain and process both semantic and non-semantic data. They lift non-semantic data to a semantic level and produce semantic output data according to its predefined model. Finally, the LW platform is open and everybody can contribute new widgets to extend its functionalities.

We illustrate the feasibility and effectiveness of the platform through location-based use cases combining data from five LOD datasets and two other open data sources. These examples can easily be extended by adding additional blocks for new datasets. Users do not have to manually browse different URIs or write SPARQL queries to retrieve and aggregate data; our approach provides a common framework to integrate and visualize the desired information.

To cope with an increasing number of blocks, we aim to investigate different kinds of automatic algorithms to present meaningful combinations of blocks to end users. Currently, we have preliminary results for block-matching and block auto-composition algorithms. These algorithms make use of the annotated input and output models of the blocks, which is a unique feature compared to similar approaches. In the future, we expect to provide users appropriately composed applications after they have formally defined their requirements, e.g., the initial input, the expected output, or the connection between them. To evaluate our platform and compare it to related approaches, we will carry out a user study. In addition, currently the developer manually has to guarantee that the output of his block adheres to the defined model. We plan to automate this process to reduce inconsistencies.

Finally, the process that allows data publishers and developers to develop blocks for each dataset should be streamlined. Automatic or semi-automatic block generation processes are therefore another important future endeavour.

## References


1. Le-Phuoc, D., Polleres, A., Hauswirth, M., Tummarello, G., Morbidoni, C.: Rapid prototyping of semantic mash-ups through semantic web pipes. In: 18th International Conference on World Wide Web, pp. 581–590. ACM, NY, USA (2009)
2. Bizer, C., Heath, T., Berners-Lee, T.: Linked data - the story so far. Int. J.Semantic Web Inf. Syst., 5(3), 1–22 (2009).

3. Hahn, R. et al.: Faceted Wikipedia Search. In: 13th International Business Information Systems Conference, pp. 1–11. Springer, Heidelberg (2010)
4. Stuhr, M., Roman, D., Norhei, D.: LODWheel - JavaScript-based Visualization of RDF Data. In: 2nd International Workshop on Consuming Linked Data. CEUR-WS.org (2011)
5. Schultz, A., Matteini, A., Isele, R., Bizer, C., Becker, C.: LDIF - Linked Data Integration Framework. In: 2nd International Workshop on Consuming Linked Data. CEUR-WS.org (2011)
6. Alahmari, F., Thom, J.A., Magee, L., Wong, W.: Evaluating Semantic Browsers for Consuming Linked Data. In: 23rd Australasian Database Conference, pp. 89–98. Australian Computer Society, Inc. (2012)
7. Matsumura, F., Kobayashi, I., Kato, F., Kamura, T., Ohmukai, I., Takeda, H.: Producing and Consuming Linked Open Data on Art with a Local Community. In: 3rd International Workshop on Consuming Linked Data. CEUR-WS.org (2012)
8. Andreas, H. et al.: On-the-fly Integration of Static and Dynamic Sources. In: 4th International Workshop on Consuming Linked Data. CEUR-WS.org (2013)
9. Endres, B., Niggemeyer: Semantic Mashups. Springer, Heidelberg (2013)
10. Trinh, T.D. et al.: Linked Widgets-An Approach to Exploit Open Government Data. In: 15th International Conference on Information Integration and Web-based Applications & Services, pp. 438–442. ACM, NY, USA (2013)
11. Trinh, T.D. et al.: Open Linked Widgets Mashup Platform. In: AI Mashup Challenge co-located with the 11st European Semantic Web Conference. CEUR-WS.org (2014)
12. Tran, T.N. et al.: Linked Data Mashups: A Review on Technologies, Applications and Challenges. In: 6th Asian Conference on Intelligent Information and Database Systems, pp. 253–262. Springer, Heidelberg (2014)
13. Bianchini, D., Antonellis. V.D.: Linked Data Services and Semantics-Enabled Mashup. Semantic Search over the Web, pp 283–307. Springer, Heidelberg (2012)
14. Auer, S. et al.: DBpedia: A Nucleus for a Web of Open Data. In: 6th International Semantic Web Conference, pp 722–735. Springer, Heidelberg (2007)
15. Krafzig, D. et al.: Enterprise SOA: Service-oriented Architecture Best Practices. Prentice Hall (2005)
16. Bizer, C. et al.: The RDF book mashup: from web APIs to a web of data. In: 3rd Workshop on Scripting for the Semantic Web. CEUR-WS.org (2007)
17. Imran, M. et al.: ResEval mash: a mashup tool for advanced research evaluation. In: 21st international conference companion on World Wide Web, pp. 361–364. ACM, NY, USA (2012)
18. Lorey, J. et al.: Black Swan: augmenting statistics with event data. In: 20th ACM international conference on Information and knowledge management, pp. 2517–2520. ACM, NY, USA (2011)
19. Cappiello, C. et al.: Dashmash: A mashup environment for end user development. In: 11th International Conference, ICWE, pp. 152-166. Springer, Heidelberg (2011)
20. Nguyen, H. et al.: Super Stream ColliderLinked Stream Mashups for Everyone. In: Semantic Web Challenge co-located with ISWC2012 (2012)
21. Jarrar, M., Dikaiakos, M.: MashQL: a query-by-diagram topping SPARQL. In: 2nd international workshop on Ontologies and information systems for the semantic web, pp. 89–96. ACM, NY, USA (2008)