Linked Widgets Platform for Rapid Collaborative Semantic Mashup Development

Tuan-Dat Trinh, Ba-Lam Do, Peter Wetz,
Peb Ruswono Aryan, Elmar Kiesling, and A Min Tjoa

TU Wien, Vienna, Austria
`{tuan.trinh,ba.do,peter.wetz,`
`peb.aryan,elmar.kiesling,a.tjoa}@tuwien.ac.at`

Abstract. In recent years, large volumes of open data have been made publicly available on the web with the explicit aim of being discoverable and usable by anyone. So far, however, it is challenging for end users to efficiently make use of open data because (i) governments and organizations do not necessarily publish their data in such a way that it can be easily discovered and understood; and (ii) there is a lack of tools that would allow non-experts to integrate open data. This paper tackles these challenges by introducing a mashup platform that combines semantic web and mashup concepts to help general users obtain insights and make informed decisions. To this end, we leverage a semantic model of mashup components for automated mechanisms that support the user in exploring available data. Moreover, we introduce a collaborative and distributed model of mashup creation and execution. This facilitates distributed ad-hoc integration of heterogeneous data contributed by multiple stakeholders.

1 Goals

Open data has the potential to create new insights and support informed decisions. Adopted by the G8 in 2013, the Open Data Charter[1] reflects the growing importance of open government data. The Charter stipulates that open data must be discoverable, accessible, and usable by all. However, as the quantity of data made available by organizations and governments grows, collecting relevant raw data from various sources, integrating it, and extracting useful information do not become easier. This is aggravated by the fact that many open data providers publish datasets without paying regard to how their data may be used or how it may be combined with data provided by others [1]. Making effective use of raw, isolated, and heterogeneous open data sets hence remains a major challenge [2].

Our goal is to address this issue and enable non-expert users to (collaboratively) integrate various open and other data sources to create new insights.

[1] `https://www.gov.uk/government/publications/open-data-charter/g8-open-data-charter-and-technical-annex` (accessed Apr. 10, 2016)

2 Proposed solution

We introduce a mashup platform built upon semantic web concepts.[2] We leverage the explicit semantics of the mashup components to ease and simplify the mashup development process, e.g., by automatically composing meaningful mashups from a given set of components. Moreover, the platform (i) facilitates collaborative work among users, data publishers, and developers to encourage widespread use of open data and (ii) integrates dispersed data stored on various devices and contributed by multiple stakeholders.

The Linked Widgets platform architecture is illustrated in Figure 1. We separate complex data integration tasks into reusable functions encapsulated in high-level user interface blocks, so-called *Linked Widgets*. These *Linked Widgets* extend the concept of standard web widgets[3] with explicit semantics. In particular, the semantics of their inputs and outputs as well as of the transformations they perform are explicitly described. Linked Widgets consume and produce Linked Data and may integrate various data from sources such as raw data in CSV, XML, JSON, or HTML format, data collected from databases, and data fetched via APIs and cloud services.

By creating a connection between an output terminal of a widget and an input terminal of another widget, users can model data flows and integrate heterogeneous data without any programming skills. Internally, we use JSON-LD[4] for the exchange of RDF data between widgets.

Based on their execution mode, widgets may be classified as *client* or *server widgets*. From a functional point of view, we can furthermore categorize them into (i) *data widgets*, (ii) *processing widgets*, and (iii) *visualization widgets*. Linked Widgets are highly reusable and can be parameterized to create mashups that span data, application logic, and presentation layers.

A web-based *collaborative mashup editor* forms the core of the data integration architecture. Multiple mashup users can compose and execute mashups simultaneously and collaboratively and integrate individually and privately contributed data with publicly available data sources. Mashup creators can locate available widgets using *semantic widget search* and group them into collections. They then connect those widgets to build mashups. The platform supports them in this process by suggesting and enforcing valid connections by means of the *terminal matching* module. In addition, the *automatic mashup composition* module can automatically compose a complete mashup from a widget, or a complete branch that consumes or provides data for a specific input or output terminal. Built on top of the *automatic mashup composition* module, the *tag-based automatic composition* module allows users to compose mashups through structured text input.

The resulting mashups fall into three categories: (i) *local mashups*, which consist exclusively of client widgets, (ii) *distributed mashups*, which consist entirely of server widgets, except for the final visualization widget(s), (iii) and

[2] `http://linkedwidgets.org/` (accessed Apr. 10, 2016)
[3] `http://www.w3.org/TR/widgets/` (accessed Apr. 10, 2016)
[4] `http://www.w3.org/TR/json-ld/` (accessed Apr. 10, 2016)

hybrid mashups, which make use of both client and server widgets. A *local mashup* does not use any resources of the platform server, because it is executed completely inside the client browser. This implies that intermediate and final data are lost as soon as the user closes the web browser. In contrast, widgets in *distributed mashups* are executed remotely as persistent applications; their output can hence be accessed at any time. *Hybrid* and *distributed mashups* can involve multiple nodes, each executing individual server widgets. This is highly useful, for instance, for streaming data scenarios where data must be collected and processed continuously.

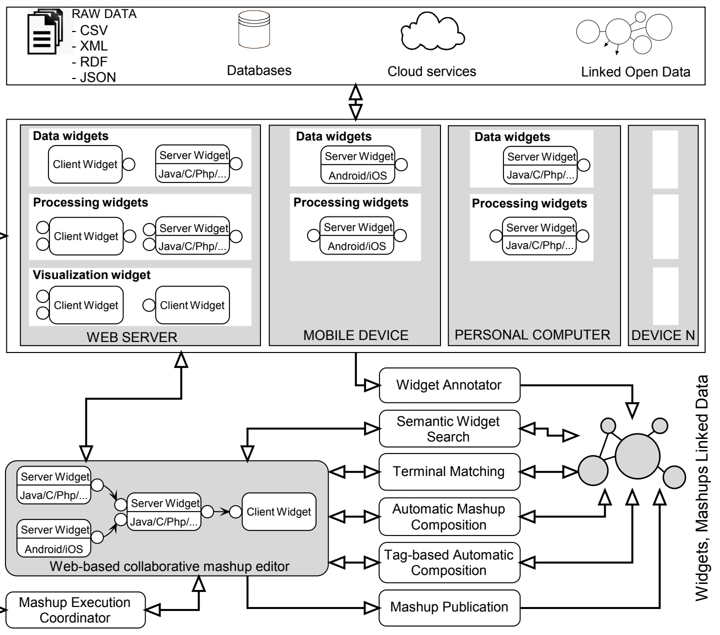


Fig. 1: Linked Widgets platform architecture

The *mashup execution coordinator* is a key component that enables widgets to cooperate. We design *local*, *remote*, and *hybrid* protocols for the three types of mashups. Based on the publish/subscribe model, the protocols facilitate efficient communication between independently developed widgets executed in various environments (e.g., browsers, Android, iOS smart phones, personal computers, or web servers) while minimizing the platform server load.

Complete mashups can by published by users on their website by means of the *mashup publication* module. A published mashup shows the final *visualization widget* only and hides all previous data processing steps from the viewer. The mashup itself can also be encapsulated as a new *data widget*.

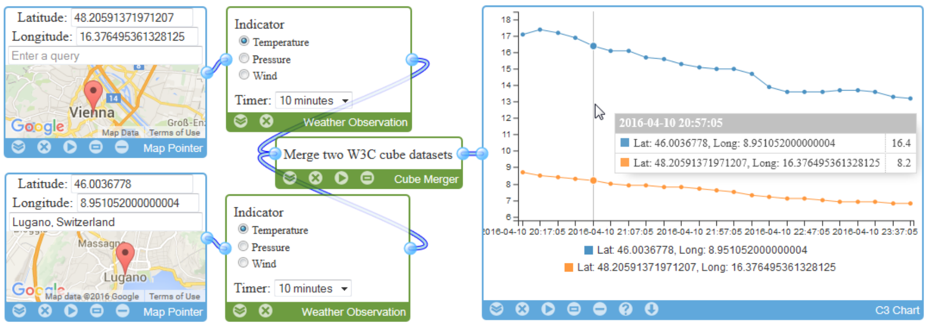
3 Maturity

A prototype implementation of the platform is complete and we are able to demonstrate it in relevant application environments.

4 Demo

Depending on available APIs, we may be able to give an interactive demo, i.e., involve the audience who may engage in the demo using their own devices and private widgets to collaboratively create a mashup and integrate data.

As a simple example of a demo application, consider the mashup depicted in Figure 2. we combine client widgets (i.e., the *Map Pointer* and the *Chart Viewer* widget) and server widgets (i.e., the *Weather Observation* and *Cube Merger* widgets) to compare weather streaming data at two locations. We first obtain the latitude and longitude of two arbitrary locations (e.g., Vienna, Austria and Lugano, Switzerland). Next, the two instances of the *Weather Observation* widget initiate two back-end services that collect the temperature at the nearest Wunderground[5] station every ten minutes. As soon as these two instances retrieve new temperature data, they convert it into a W3C cube dataset[6] format (with two dimensions, *location* and *observation time*, and one measure, i.e., *temperature*) to the *Cube Merger* widget; this widget aggregates the new data with the data it has collected since the inception of the mashup. Finally, the *Chart Viewer* visualizes the aggregated W3C cube dataset. Because the *Weather Observation* widget provides new output data every ten minutes and because both the *Weather Observation* and the *Cube Merger* are server widgets, this example mashup is a persistent streaming mashup (i.e., the mashup will keep running in the back-end and present the complete observations data collected whenever it is reopened).



Latitude: 48.20591371971207
Longitude: 16.376495361328125
Enter a query

Latitude: 46.0036778
Longitude: 8.951052000000004
Lugano, Switzerland

Indicator: [✓ Temperature] Pressure Wind
Timer: 10 minutes

Indicator: [✓ Temperature] Pressure Wind
Timer: 10 minutes

Merge two W3C cube datasets

2016-04-10 20:57:05
🟦 Lat: 46.0036778, Long: 8.951052000000004 16.4
🟧 Lat: 48.20591371971207, Long: 16.376495361328125 8.2

Fig. 2: Vienna and Lugano weather comparison

References

1. Braunschweig, K., Eberius, J., Thiele, M., Lehner, W.: The State of Open Data - Limits of Current Open Data Platforms. In: Proceedings of the 21st World Wide Web Conference 2012, Web Science Track at WWW '12, Lyon, France, April 16-20, 2012. ACM (2012)
2. Jaakkola, H., Mäkinen, T., Eteläaho, A.: Open Data: Opportunities and Challenges. In: Proceedings of the 15th International Conference on Computer Systems and Technologies. pp. 25–39. CompSysTech '14, ACM, New York, NY, USA (2014)

[5] `https://www.wunderground.com/` (accessed Apr. 10, 2016)
[6] `https://www.w3.org/TR/vocab-data-cube/` (accessed Apr. 10, 2016)

A Mashup Feature Checklist

Mashup Type
- Data mashups
- Logic mashups
- User Interface (UI) mashups
- **Hybrid mashups**

Component Types
- **Data components**
- **Logic components**
- **UI components**

Runtime Location
- Client-side only
- Server-side only
- **Both Client and Server**

Integration Logic
- UI-based integration
- **Orchestrated integration (local and hybrid mashups)**
- **Choreographed integration (distributed server-side mashups)**

Instantiation Lifecycle
- Stateless
- **Short-living**
- **Long-living (persistent mashups)**

B Mashup Tool Feature Checklist

Targeted End-User
- Local Developers
- **Non Programmers**
- Expert Programmers

Automation Degree
- **Full Automation**
- **Semi-automation**
- **Manual**

Liveness Level
- Level 1 (Non-Executable Prototype Mockup)
- Level 2 (Explicit Compilation and Deployment Steps)
- **Level 3 (Automatic Compilation and Deployment, requires Re-initialization)**

– Level 4 (Dynamic Modification of Running Mashup)

Interaction Technique
– Editable Example
– Form-based
– Programming by Demonstration
– Spreadsheets
– Textual DSL
– Visual Language (Iconic)
– **Visual Language (Wiring, Implicit Control Flow)**
– Visual Language (Wiring, Explicit Control Flow)
– WYSIWYG
– Natural Language
– Other

Online User Community
– None
– Private
– **Public**