

# Linked Data-driven Web Components

Ali Khalili  
Dept. of Computer Science  
VU University Amsterdam  
The Netherlands  
a.khalili@vu.nl

Antonis Loizou  
Dept. of Computer Science  
VU University Amsterdam  
The Netherlands  
a.loizou@vu.nl

Frank van Harmelen  
Dept. of Computer Science  
VU University Amsterdam  
The Netherlands  
frank.van.harmelen@vu.nl

## ABSTRACT

importance of web components. Linked Data app development is cumbersome. combining the idea to create LD-R. advantage of approach briefly.

## Categories and Subject Descriptors

D.2.13 [Software Engineering]: Reusable Software

## General Terms

Design, Human Factors, Standardization

## 1. INTRODUCTION

general about linked data applications and web components

*Web Components* are a set of W3C standards that enable the creation of reusable widgets or components in Web documents and Web applications. Web components aim to bring *Component-Based Software Development* (CBSD) to the World Wide Web. Some advantages of CBSD approach are reusability, replacability, extensibility, encapsulation and independence.

introducing LD-R

Ld-R offers many benefits that we will describe in the remainder of the paper. Among them are: - -

## 2. CONTRIBUTIONS AND OUTLINE

The contributions of this work are...

We evaluate this claim by...

We explore these claims in stages...

## 3. LINKED DATA-DRIVEN WEB COMPONENTS

We define a *Linked Data-driven* (LD-R) Web Component as a Web component that employs RDF data model for representing its content and specification (i.e. metadata about the component).

### 3.1 Features

Linked Data-driven Web components provide the following features:

- *Fine-grained Web applications.* Resource Description Framework (RDF) provides a common data model that allows data-driven components to be created, shared and integrated in a structured way across different applications. Figure 1 depicts the 4 main component levels in a Linked Data-driven Web application. The dataflow in the application starts from the *Dataset* component which handles all the events related to a set of resources embedded in a named graph. The next level is the *Resource* component which is identified by a URI and indicates what is described in the application. A resource is specified by a set of properties which are handled by the *Property* component. Properties can be either individual or aggregate when combining multiple features of a resource (e.g. a component that combines longitude and latitude properties; start date and end date properties for a date range, etc.). Each property is instantiated by an individual value or multiple values in case of an aggregate object. The value(s) of properties are controlled by the *Object* component. Object component invokes different components to view, edit and browse the property values. *Viewer*, *Editor* and *Browser* components are terminals in the LD-R single directional data flow where customized user-generated components can be plugged into the system. These components apply on individual and aggregate objects (e.g. to show multiple coordinates on a the map).

In addition to the fine-grained component architecture, LD-R Web applications provide a fine-grained access control over the data provided by the components. RDF-based access control in LD-R applications operates at four different granularities provided by Dataset, Resource, Property and Object component levels. For example, we can restrict access to a specific property of a specific resource in a certain dataset.

- *Customization and Personalization.* LD-R provide a versatile approach for context adaptation. A context

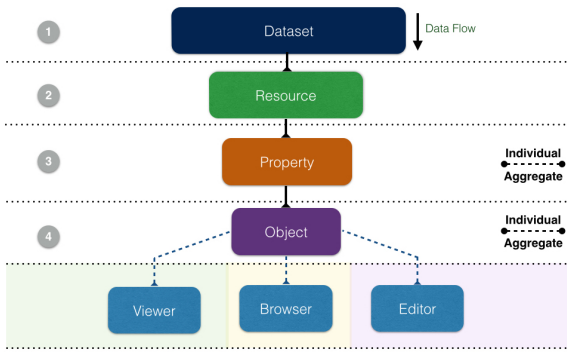


Figure 1: Architecture of LD-R Applications.

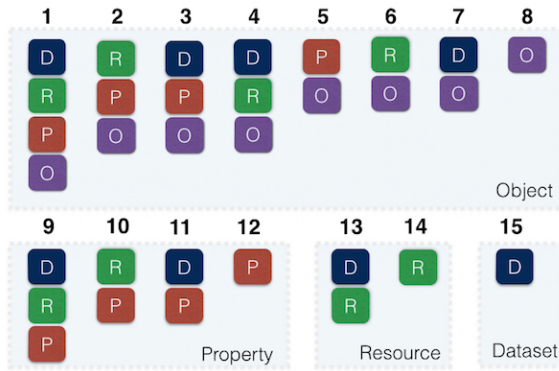


Figure 2: LD-R Scopes.

can be a specific domain of interest, a specific user requirements or both. In order to enable customization and personalization, LD-R exploits the concept of *Scope*. A scope is defined as a directed combination of Dataset, Resource, Property and Object components (cf. Figure 2). Each scope conveys a certain level of specificity on a given context ranging from 1 (most specific) to 15 (least specific). Scopes are defined by using the URIs for RDF resources and types. For example, on the property level, we can define a generic configuration for all properties and then for some specific properties (e.g. `dcterms:title`, `rdfs:label`) within a specific resource (e.g. `<http://ld-r.org>`), we can change or overwrite those configurations.

Scopes can also be defined under a specific user which facilitates versioning and reuse of user-specific configs. User-specific configs provide different views on components and thereby data, based on the different personas dealing with those components and data.

- *Component/Content Visibility and Reusability.*  
- RDFa, Microdata

### 3.2 Life Cycle

As shown in Figure 3, the LD-R components lifecycle encompasses four primary types of stakeholders:

- *Linked Data Provider.* Since the LD-R approach focuses mainly on Linked Data applications, provision

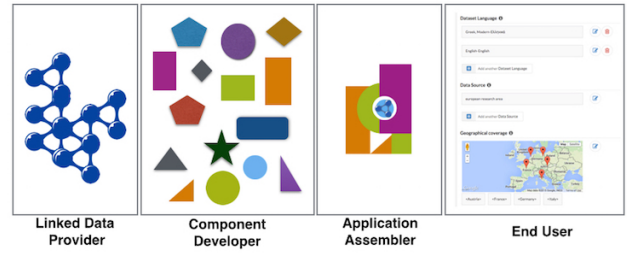


Figure 3: LD-R Components Life Cycle.

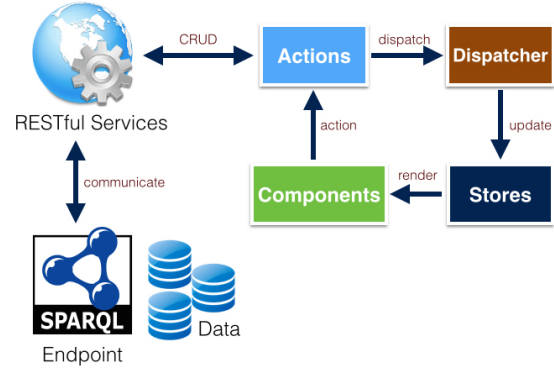


Figure 4: Data Flow

of RDF-compliant data is an essential phase in developing the LD-R components. \*Data Scientists and different steps in providing data from LOD2 project

- *Component Developer.* It includes programmers who are involved in component fabrication.
- *Application Assembler.* The main task of application assembler is to identify the right components and configurations for the application; and combine them in a way which fits the application requirement.
- *End User.* It is the user who experiences working with components to pursue his goals on a certain application domain. The end user is the one who requests developing a component and the one who sends feedback on the existing components.

## 4. IMPLEMENTATION

<http://ld-r.org>

A monolithic application puts all its functionality into a single process and scales by replicating the monolith on multiple servers. A microservices architecture puts each element of functionality into a separate service and scales by distributing these services across servers, replicating as needed [2].

## 5. EVALUATION

RISIS

OpenPhacts

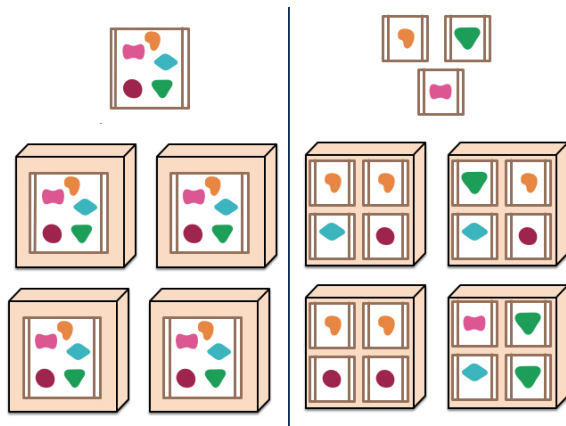


Figure 5: Monoliths vs. Microservices [2]

[2] J. Lewis and M. Fowler. Microservices, 2014. <http://martinfowler.com/articles/microservices.html>.

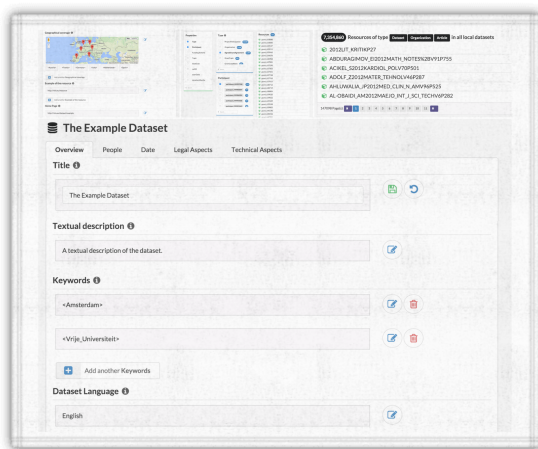


Figure 6: Screenshot

## 6. DISCUSSION

## 7. RELATED WORK

Web Components and the Semantic Web [1]

Semantic Web Services

Existing tools to view/edit and browse LD e.g. OntoWiki, Saha

## 8. CONCLUSION

## 9. ACKNOWLEDGEMENT

We would like to thank our colleagues from the KRR research group at VU University Amsterdam for their helpful comments during the development of the LD-R framework. This work was supported by a grant from the European Union's 7th Framework Programme provided for the project RISIS (GA no. 313082).

## 10. REFERENCES

[1] M. Casey and C. Pahl. Web components and the semantic web. *Electr. Notes Theor. Comput. Sci.*, 82(5):156–163, 2003.