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

With the recent advent of Web Components, WWW is moving towards becoming a rich ecosystem of discoverable, sharable, and reusable data-driven components. As the amount of components increase on the Web, identifying the right components and accessing the data encapsualted within these components become a challenging issue. On the other hand, Semantic Web with the goal of understanding data on the Web has gained traction in recent years and more semantic data has become available online. Semantic Web and Linked Data standards provide support for richer data discovery, integration and navigation on the Web. Linked Data-driven (LD-R) Web Components is an attempt to combine best practices for data integration from Linked Data with the development of Web components for component integration. In this paper, we discuss the idea of semantically enhananced Web components. In order to show the feasibility of our approach, we also present an open source software framework which enables creation and integration of LD-R Web Components.

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 reusibility, replacability, extensibility, encapsulation and independence.

W3C specifications of Web Components [2]

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 datae and end date properties for a date range, etc.). Each property is instantiated by an individual value or mutiple values in case of an aggreagte object. The value(s) of properties are controled by the *Object* component. Object component invokes different components to view, edit and browse the prop-

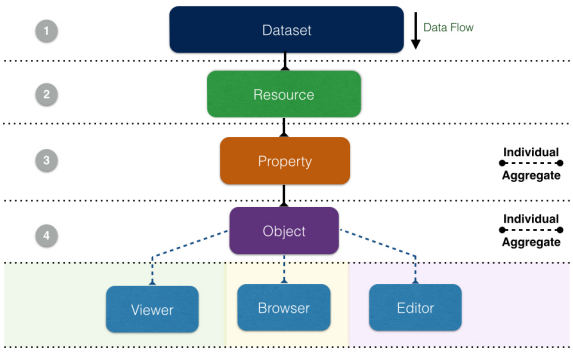

Figure 1: Architecture of LD-R Applications.

Figure 2: LD-R Scopes.

erty 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 can be a specific domain of interest, a specific user requirments or both. In order to enable customization and personalization, LD-R exploits the concept of *Scope*. A scope is defiened 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 specfic). 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-specifc configs provide different views on components and thereby data, based on the different personas dealing with those components and data.

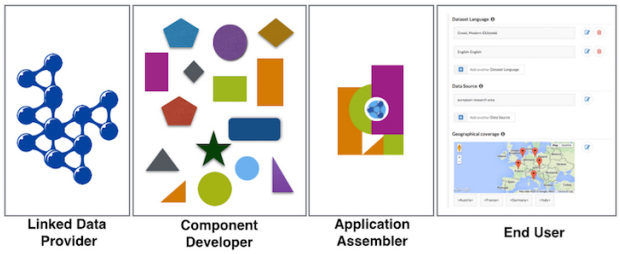

Figure 3: LD-R Components Life Cycle.

- *Component Visibility and Reusability.*

  metadata about components (https://github.com/ali1k/ld-r-metadata-generator) in JSON-LD

  general metadata: name, description, version, homepage, author, etc.

  specific metadata: level, granularity (individual, aggregate), mode (view, edit, browse), dependencies (internal, external), config parameters with description

  use Schema.org SoftwareApplication schema.

- *Content Visibility and Reusability.*

  Component content represented in RDFa, Microdata. exmaple: good relations for online shopping and SEO

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

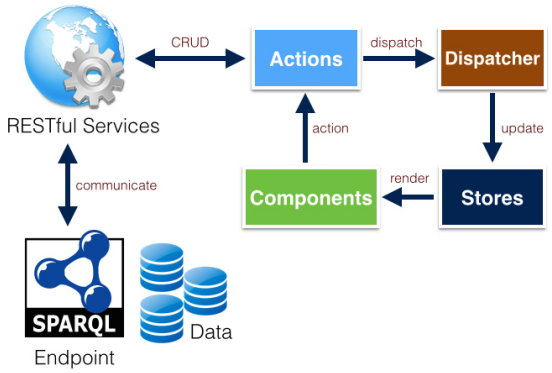

Figure 4: Data Flow in LD-R framework.

4. IMPLEMENTATION

In order to realize the idea of Linked Data-driven Web components, we implemented a software framework called *Linked Data Reactor (LD-R)* which is aviable online at `http://ld-r.org`. LD-R utilizes Facebook's ReactJS[1] components and Flux[2] architecture, Yahoo!'s Fluxible[3] framework for isomorphic Web applications and Semantic-UI[4] framework for flexible UI themes.

The main reasons we chose *React* components over other existing solutions (e.g. Polymer[5], AngularJS[6], EmberJS[7], etc.) were the maturity of the technology, maintainablity, number of developer tools/components/applications, and efficiency[8]. As shown in Figure 4, LD-R follows the Flux architecture which eschews MVC (Model-View-Controller) in favor of a unidirectional data flow. When a user interacts with a React component, the component propagates an action through a central dispatcher, to the various stores that hold the application's data and business logic, which updates all of the components that are affected.

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 separete service and scales by distributing these services across servers, replicating as needed [3].

5. EVALUATION

RISIS

OpenPhacts

6. DISCUSSION

7. RELATED WORK

[1]`https://facebook.github.io/react/`
[2]`https://facebook.github.io/flux`
[3]`http://fluxible.io/`
[4]`http://semantic-ui.com/`
[5]`http://www.polymer-project.org/`
[6]`https://angularjs.org/`
[7]`http://emberjs.com/`
[8]Elaborating on all these factors is beyond the scope of this paper.

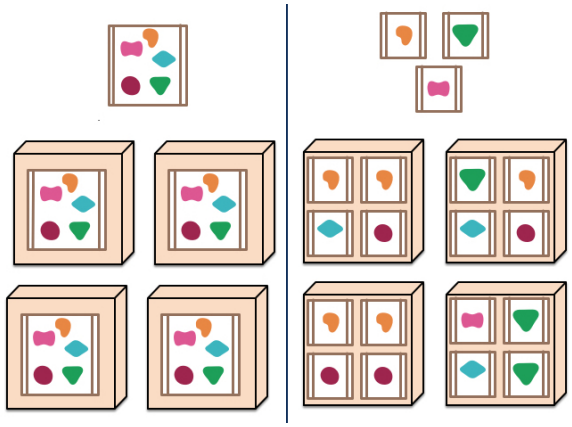

Figure 5: Monoliths vs. Microservices [3]

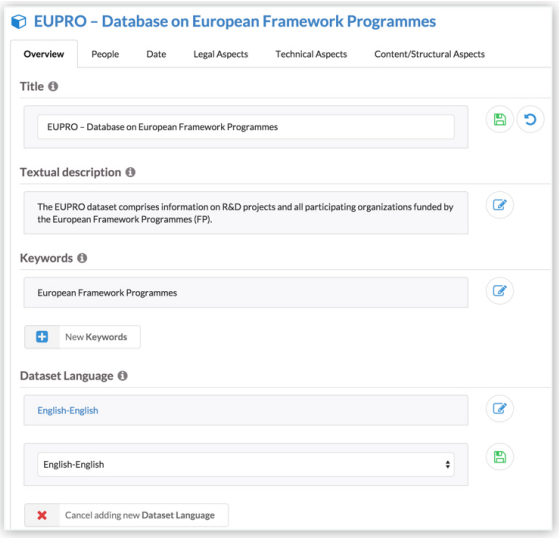

Figure 6: Screenshot

Web Components and the Semantic Web [1]

Semantic Web Services

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

8. CONCLUSION

LD-R approach not only facilitates the discovery and reuse of Web components but also makes the creation of Linked Data application easier.

9. AKNOWLEDGEMENT

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 EuropeanÂ UnionâĂŹs 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.

[2] D. Cooney. Introduction to web components, 2014. `http://www.w3.org/TR/components-intro/`.

[3] J. Lewis and M. Fowler. Microservices, 2014. `http://martinfowler.com/articles/microservices.html`.