Adaptive Linked Data-driven Web Components: Building Flexible and Reusable Semantic Web Interfaces

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
The amount of published Linked Data on the Web is increasing day by day. As a result, the applications driven by Semantic Web and Linked Data are taking momentum on the Web. One of the major entrance barriers for Web developers to contribute to this wave of Linked Data Applications (LDAs) is the required knowledge of Semantic Web technologies such as RDF data model and SPARQL query language to interact with the triple stores. This paper presents an adaptive component-based approach for creating flexible and resuable Semantic Web interfaces driven by Linked Data. Linked Data-driven (LD-R) Web components abstract the complexity of underlying Semantic Web technologies in order to allow reuse of existing Web components in LDAs and to enable Web developers who are not expert in Semantic Web to develop interfaces to view, edit and browse Linked Data. In addition to modularity provided by LD-R components, the proposed RDF-based configuration method allows application assemblers to reshape their user interface for different use cases, by either reusing existing shared configs or by creating their proprietary confgis.

Categories and Subject Descriptors
D.2.13 [**Software Engineering**]: Reusable Software

General Terms
Design, Human Factors, Standardization

1. INTRODUCTION
With the growing number of structured data published on the Web, WWW is moving towards becoming a rich ecosystem of machine-understandable Linked Data (LD) [1]. Semantically structured data facilitate a number of important aspects of information management such as information retrieval, search, visualization, customization, personalization and integration [4]. Despite all these benefits, Linked Data Applications (LDAs) have not yet grasped well by the large community of Web developers outside the Semantic Web domain and causally, by the end users on the Web. The usage of semantic data is still quite limited and most of the currently published Linked Data are generated by a relatively small amount of publishers [3] which indicate some entrance barriers towards wide-spread utilization of Linked Data [7].

[1] `lodlaundromat.org` recently (25.09.2015) reported approx. 38.6 billion triples published on the Web.

The current communication gap between Semantic Web developers and User Experience (UX) designers, driven by the need to bear Semantic Web knowledge, prevents streamline flow of best practices from UX community into the Linked Data user interfaces (UIs). The resulting lack of adoption and standardization makes current LDAs not often function consistent with user expectations and impels more development time and costs on LDA deveopers. In this situation, more time is spent in re-designing existing UIs rather than focusing on innovation and creation of sophisticated LDAs.

This paper presents adaptive Linked Data-driven Web components as an approach to build flexible and reusable Semantic Web UIs. *Web Components* are a set of W3C standards [2] that enable the creation of reusable widgets or components in Web documents and Web applications. Linked Data-driven (LD-R) Web components as defiend in this paper are a species of Web components that employ RDF data model for representing their content and specification (i.e. metadata about the component). LD-R components are supported by a set of predefined core Web components each representing a compartment of the RDF data model on the Web UI. LD-R components enable encapsualtion of the Semantic Web nature of an LDA thereby allow UX designers and Web developers outside the Semantic Web community to contribute to LDAs. They also allow current Semantic Web developers to reuse existing Web components in their LDAs. Furthermore, LD-R components exploit the power and flexibility of RDF data model in describing and sharing resources to provide a mechanism to adapt the Web interfaces based on the meaning of data and user-defined rules.

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

Bootstrapping LDA UIs. LD-R components exploit best practices from modern Web application development to bring an exhaustive architecture to perform separation of concerns and thereby bootstrapping an LDA by only selecting a minimal relavant configuration. For example, a developer only needs to set the URL of his in-use SPARQL endpoint and start developing the viewer components without dealing with the underlying connection adapters and data flow mediators in the system.

Standardization and Reusability of LDA UIs. *write about the advatnages of CBSD

Customization and Personalization of LDA UIs. *wrtie about RDF-based config

Adoption of LDA UIs by non-Semantic Web developers. *write about Skeumorphic design, affordance [6]

2. CONTRIBUTIONS AND OUTLINE

The contributions of this work are the concept of *Adaptive LD-R Web components* and an open source implementation of it available at `http://ld-r.org`. Our primary claim is that adopting a component-based approach that encapsualtes the main concerns of a Semantic Web application, paves the way to reuse existing best practices from the UX community within the LDAs hence building more usable and pervasive LDAs.

We explore this claim in stages. First, we collect some data about the current status of Semantic Web UI development. Next, we demonstrate how adaptive LD-R Web components can address the current issues in LDA UI development. Finally, we discuss the implementation of our idea and its usage in real-world scenarios.

3. THE CURRENT STATUS OF SEMANTIC WEB USER INTERFACE DEVELOPMENT

addressing Semantic Web developers on Github. used embedded form, less questions to attract more people!

7 questions to address the following topics: - developer experience and proffiency - need for a frameowrk to bootstrap building LDA interfaces - need for reusing code - need for customization of interface - adoption of current Web components - adoption barriers for non-SW developers to work on LDAs

4. COMPONENT-BASED DEVELOPMENT OF LINKED DATA APPLICATIONS

*monolith vs microservices

4.1 Separation of Concerns

*categorize technical aspects

*need for a framework to bootstrap LDA UIs, talk about different concerns in LDAs already implemented in LD-R

4.2 Linked Data-driven Web Components

* component architecture

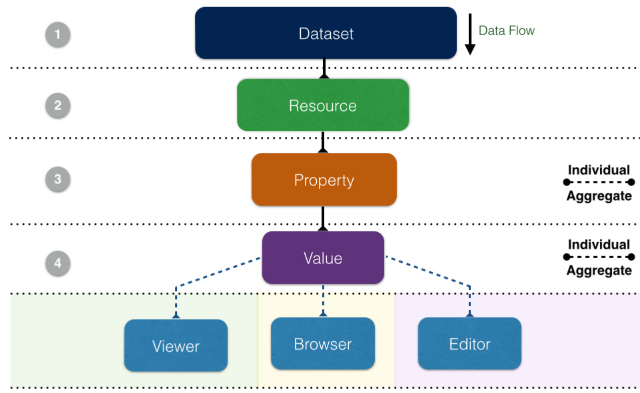

Figure 1: Architecture of LD-R Applications.

*Web Components and how they relate to SW, content and metadata

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 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.

4.3 User Interface Adaptation

*RDF-based config

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,

Figure 2: LD-R Scopes.

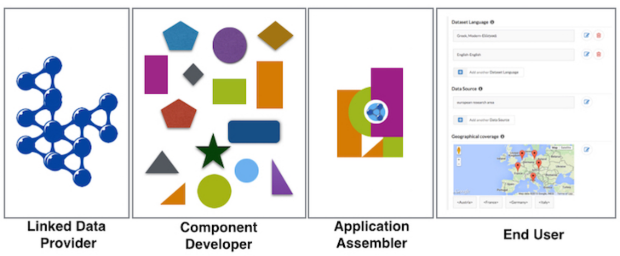

Figure 3: LD-R Components Life Cycle.

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.

4.4 Stackeholders and Life Cycle

5. THE LIFE CYCLE OF LD-R WEB COMPONENTS

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.

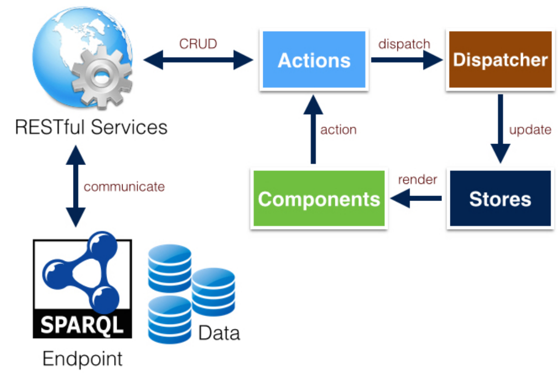

Figure 4: Data Flow in LD-R framework.

- *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.

6. IMPLEMENTATION

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

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.

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 available online at `http://ld-r.org`. LD-R utilizes Facebook's ReactJS[2] components and Flux[3] architecture, Yahoo!'s Fluxible[4] framework for isomorphic Web applications and Semantic-UI[5] framework for flexible UI themes.

The main reasons we chose *React* components over other existing solutions (e.g. Polymer[6], AngularJS[7], EmberJS[8], etc.) were the maturity of the technology, maintainablity,

[2]`https://facebook.github.io/react/`
[3]`https://facebook.github.io/flux`
[4]`http://fluxible.io/`
[5]`http://semantic-ui.com/`
[6]`http://www.polymer-project.org/`
[7]`https://angularjs.org/`
[8]`http://emberjs.com/`

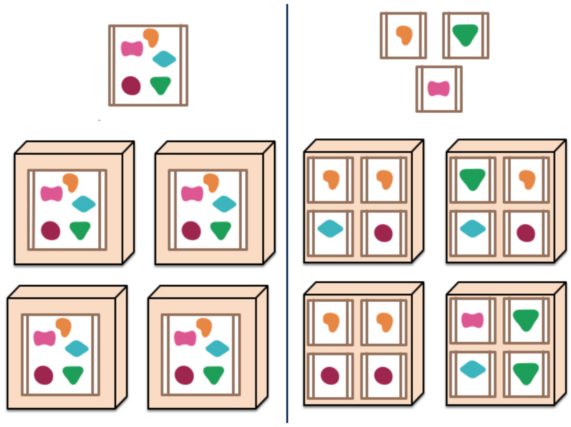

Figure 5: Monoliths vs. Microservices [5]

number of developer tools/components/applications, and efficiency[9]. 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. The component interaction with SPARQL endpoints to retrieve and update Linked Data occurs via invoking RESTful services in actions.

In contrast to the centalized monolithic architecture, LD-R components comply with *Microservices Architecture* [5]. As shown in Figure 5, microservices architecture puts the main functionalities of a component into separete services (instead of in-memory function calls) and scales by distributing these services across servers, replicating as needed. This architectural style also minimizes the redeploying of the entire application when changes in components occure.

7. EVALUATION

two types of evaluations:

2. evaluating the solution provided by us: RISIS OpenPhacts

8. DISCUSSION

9. RELATED WORK

Web Components and the Semantic Web [1]

Semantic Web Services

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

10. CONCLUSION

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. LD-R approach

[9]Elaborating on all these factors is beyond the scope of this paper.

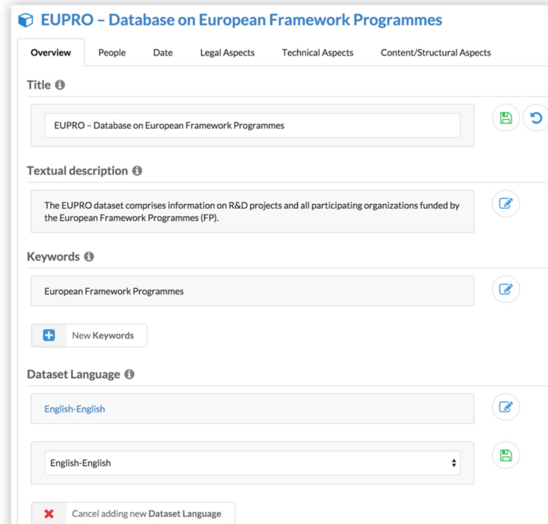

Figure 6: Screenshot

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

11. 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 European Union's 7th Framework Programme provided for the project RISIS (GA no. 313082).

12. 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] P. Frischmuth, M. Martin, S. Tramp, T. Riechert, and S. Auer. OntoWiki—An Authoring, Publication and Visualization Interface for the Data Web. *Semantic Web Journal*, 2014.

[4] A. Khalili and S. Auer. User interfaces for semantic authoring of textual content: A systematic literature review. *Web Semantics: Science, Services and Agents on the World Wide Web*, 22(0):1 – 18, 2013.

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

[6] D. A. Norman. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, Inc., New York, NY, USA, 2013.

[7] T. Stegemann and J. Ziegler. Semwidgjs: A semantic widget library for the rapid development of user interfaces for linked open data. In *44. Jahrestagung der Gesellschaft für Informatik, Informatik 2014, Big Data - Komplexität meistern, 22.-26. September 2014 in Stuttgart, Deutschland*, pages 479–490, 2014.