Linked Data Reactor: Towards Data-aware User Interfaces

Ali Khalili
Department of Computer Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
a.khalili@vu.nl

Klaas Andries de Graaf
Department of Computer Science
Vrije Universiteit Amsterdam
Amsterdam, the Netherlands
ka.de.graaf@vu.nl

ABSTRACT

Most of the existing Web user interfaces (UIs) are hard-coded by their developers to address certain predefined types of data, and hence are blind to the semantics of data they are dealing with. When talking about unstructured data or data without an explicit semantic representation, our expectations of data-awareness are lower. However, when we consider Linked Data UIs, where we have both structured data and semantics, we indeed expect more awareness from the UI which renders the data. In this paper we present an architecture for data-aware UIs, called Linked Data Reactor, implemented based on Web components and Semantic Web technologies. The proposed UIs can understand users' data and are capable to interact with users accordingly.

Keywords

Semantic Web, Linked Data, User Interfaces, Adaptation

1. INTRODUCTION

Most of the existing Web user interfaces (UIs) are hard-coded by their developers to address certain predefined types of data, hence are blind to the semantics of data they are dealing with. When talking about unstructured data or data without an explicit semantic representation, our expectations of data-awareness are lower. However, when we consider Linked Data UIs where we have both data and semantics, we indeed expect more awareness from the UI which renders the data. For example, Figure 1 shows DBpedia pages for three totally different entity types (Apple as a Fruit, Apple as a Company and Amsterdam as a City) which are presented with the same generic template, completely ignoring the meaning of the underlying data. In this case, it does not matter what the data user thinks or what the underlying data means, what matters is what the application developer thinks! A user might instead want to see Apple's shareholders and its stock status, or see the health benefits of apples, or, being a farmer, see information on how to grow apples. A relevant question here is *why UIs can understand data but do not interact with users accordingly?* It is because there is a gap between what UIs provide (functionality) and what the meaning of data is (semantics). Making a bridge between the emerging two worlds of Web Components and Semantic Web technologies is an approach to support UIs that understand users' data and are capable to interact with users accordingly. In [6], we conducted a survey targeting 79 active Semantic Web developers to ask them about the current pitfalls in developing Semantic Web UIs. Based on the results, developers spend a lot of time (on average more than 2 days) on bootstrapping their applications before they can start working on the UI. A considerable amount of users (46%), prefer to write the code from scratch instead of reusing code from existing Semantic Web projects and 52% had experience of manually adapting the user interface of their applications frequently. To address those concerns, in this paper we present an architecture for data-aware UIs implemented in a software framework called *Linked Data Reactor* (LD-R).

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2. ARCHITECTURE

Figure 2 depicts our proposed architecture for data-aware UIs where related elements are color coded. The system provides three main modes of interaction with data namely view, browse, and edit. The system adapts its behavior during user interactions by providing appropriate interactive UI components based on the semantics of data and the given user context. In the following sections we describe the main building blocks of the architecture.

2.1 Interaction Layer

LD-R exploits four core component levels (i.e., reactor components) to abstract the actions required for retrieving and updating the graph-based data and provides a basis for user-defined components to interact with Linked Data in three modes: view, edit and browse. The data-flow in the system starts from the Dataset component which handles all the events related to a set of resources under a named graph identified by a URI. The next level is the Resource component which is identified by a URI and indicates what is described in the application. A resource is described 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

About: Apple Inc.
About: Apple
About: Amsterdam
http://dbpedia.org/page/Apple_Inc.
http://dbpedia.org/page/Apple
http://dbpedia.org/page/Amsterdam

Figure 1: Same presentation of data for totally different entity types on DBpedia.

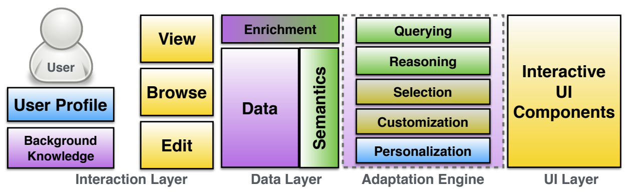


Figure 2: The proposed architecture for data-aware UIs implemented in the LD-R framework.

controlled by the Value component. In turn, Value components invoke different components to view, edit and browse the property values. Also see Figure 4.

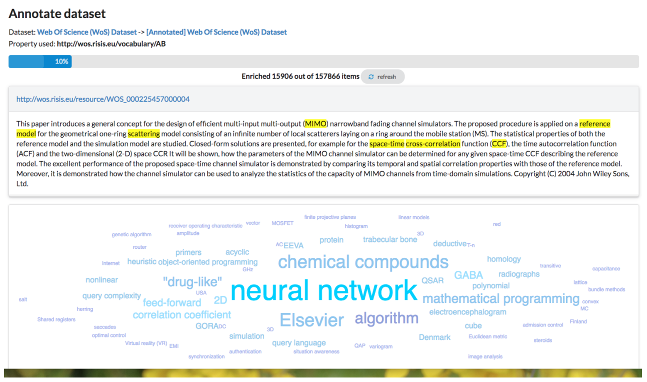
Viewer, Editor and Browser components are terminals in the LD-R single directional data flow where customized user-generated components can be plugged into the system. For instance, the following components are already built-in to the system[1]:

- *Viewer*: ImageView, MapView, OptionView, LanguageView, DBpediaView, YASQEView
- *Browser*: TagList, GeoList, ChartList, GeoList
- *Editor*: CalendarInput, DBpediaInput, LanguageInput, PrefixInput, OptionInput

2.2 Data Layer

There are five different sorts of data taken into account within the LD-R environment: 1) *user's profile data* to understand the user preferences, 2) *user's background knowledge* to consider a user's domain of interest while browsing data, 3) *original data* to be browsed, 4) *configuration data* as output of adaptation process to customize and personalize both data and UIs, 5) *complementary data* added as enrichment to original data for richer contextualization. All the above datasets are represented as single or multiple RDF graphs) to be ready for integration (using federated SPARQL queries) and analysis.

[1]full list of components is available at http://demo.ld-r.org/documentation



Annotate dataset

Dataset: Web Of Science (WoS) Dataset -> [Annotated] Web Of Science (WoS) Dataset
Property used: http://wos.risis.eu/vocabulary/AB

10%

Enriched 15906 out of 157866 items refresh

http://wos.risis.eu/resource/WOS_000225457000004

This paper introduces a general concept for the design of efficient multi-input multi-output (MIMO) narrowband fading channel simulators. The proposed procedure is applied on a reference model for the geometrical one-ring scattering model consisting of an infinite number of local scatterers laying on a ring around the mobile station (MS). The statistical properties of both the reference model and the simulation model are studied. Closed-form solutions are presented, for example for the space-time cross-correlation function (CCF), the time autocorrelation function (ACF) and the two-dimensional (2-D) space CCR It will be shown, how the parameters of the MIMO channel simulator can be determined for any given space-time CCF describing the reference model. The excellent performance of the proposed space-time channel simulator is demonstrated by comparing its temporal and spatial correlation properties with those of the reference model. Moreover, it is demonstrated how the channel simulator can be used to analyze the statistics of the capacity of MIMO channels from time-domain simulations. Copyright (C) 2004 John Wiley Sons, Ltd.

Figure 3: Interactive Content Annotation in LD-R.

LD-R supports resource annotation to interlink the original data with the user's background knowledge and to generate complementary data connected to the original data to be browsed. At the moment, two types of annotation are supported within the system: *Named Entity Recognition* (NER) using DBpedia Spotlight[2] and *Geo-boundary-tagging* supported by open geo boundaries from OpenStreetMap and GADM[3]. As shown in Figure 3, there are interactive UIs embedded in the LD-R system to annotate a dataset while progressively communicating the results of annotation.

Another feature which is supported out-of-the-box in LD-R is embedding RDFa and Microdata annotations for external applications such as search engines to extract structured data from LD-R powered UI components. For example, an LD-R component created based on the *Good Relations* or Schema.org ontologies, can automatically expose the product data as Google Rich Snippets, which will provide better visibility of the data on Web search results (i.e. SEO).

2.3 Adaptation Engine

An adaptive UI[4] is a UI which adapts, that is, changes its layout and elements to the needs of the user or context and is similarly alterable by each user. LD-R incorporates an

[2]http://www.dbpedia-spotlight.org
[3]http://gadm.org
[4]http://en.wikipedia.org/wiki/Adaptive_user_interface

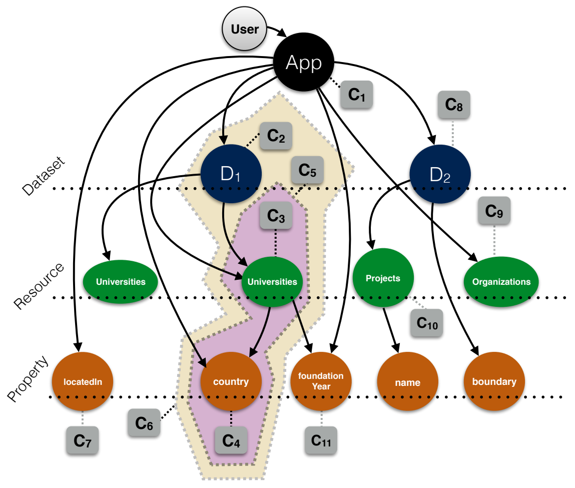
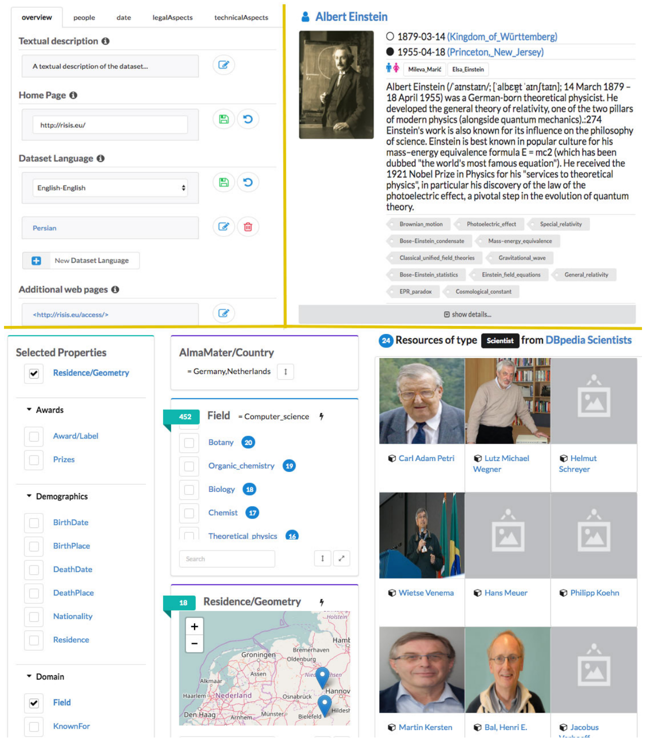


Figure 4: An example configuration hypergraph.

adaptation engine to realize data-aware UIs when users interact with data. The task of adaptation engine is to make a bridge between data (enriched by semantics) and existing UI components suitable to render data. The adaptation engine includes the following core components:

- *Querying.* This part is responsible for composing, sharing and running of SPARQL queries within the LD-R environment. The queries are used to retrieve manual user-driven configurations.
- *Reasoning.* This is the core part of the engine where different datasets mentioned in Section 2.2 are analyzed in an integrative way to find the best strategy for data rendering and UI augmentation.
- *Selection.* This part allows to manually or automatically, as a result of reasoning, select or replace and existing UI component.
- *Customization.* This part allows to manually or automatically customize an existing UI component.
- *Personalization.* This part allows to manually or automatically personalize an existing UI component. Personalization will overwrite the configurations used for customization to consider the user's context.

Figure 4 shows that the configuration process is done by traversing the hypergraph generated either manually by a user or automatically as result of reasoning. LD-R exploits a hierarchical permutation of the Dataset, Resource, Property, and Value (DRPV) components as *scopes* to select specific parts of the UI to be customized or personalized. Each scope conveys a certain level of specificity on a given context ranging from 1 (most specific: DRPV) to 15 (least specific: D (Dataset). Scopes are defined by using either the URIs of named graphs, resources, and properties, or by identifying the resource types and data types. A configuration is defined as a setting which affects the way the UI components are interpreted and rendered (e.g., render a specific component for a specific RDF property or a specific RDF resource within a specific RDF graph). UI adaptation is handled by



Textual description
A textual description of the dataset...
Home Page
http://risis.eu/
Dataset Language
English-English
Persian
New Dataset Language
Additional web pages
<http://risis.eu/access/>

Albert Einstein
1879-03-14 (Kingdom_of_Württemberg)
1955-04-18 (Princeton,_New_Jersey)
Mileva_Marić Elsa_Einstein

Albert Einstein (/ˈaɪnstaɪn/; [ˈalbɛʁt ˈaɪnʃtaɪn]; 14 March 1879 – 18 April 1955) was a German-born theoretical physicist. He developed the general theory of relativity, one of the two pillars of modern physics (alongside quantum mechanics).:274 Einstein's work is also known for its influence on the philosophy of science. Einstein is best known in popular culture for his mass–energy equivalence formula E = mc2 (which has been dubbed "the world's most famous equation"). He received the 1921 Nobel Prize in Physics for his "services to theoretical physics", in particular his discovery of the law of the photoelectric effect, a pivotal step in the evolution of quantum theory.

Brownian_motion, Photoelectric_effect, Special_relativity, Bose-Einstein_condensate, Mass-energy_equivalence, Classical_unified_field_theories, Gravitational_wave, Bose-Einstein_statistics, Einstein_field_equations, General_relativity, EPR_paradox, Cosmological_constant

Selected Properties
[✓] Residence/Geometry
Award/Label, Prizes
BirthDate, BirthPlace, DeathDate, DeathPlace, Nationality, Residence
[✓] Field, KnownFor

AlmaMater/Country = Germany,Netherlands
452 Field = Computer_science
Botany 20
Organic_chemistry 19
Biology 18
Chemist 17
Theoretical_physics 16
18 Residence/Geometry

24 Resources of type Scientist from DBpedia Scientists
Carl Adam Petri, Lutz Michael Wegner, Helmut Schreyer, Wietse Venema, Hans Meuer, Philipp Koehn, Martin Kersten, Bal, Henri E., Jacobus

Figure 5: Edit, View and Browse mode in the LD-R.

traversing the configurations for scopes, populating the configurations and overwriting the configurations when a more specific applicable scope is found.

3. IMPLEMENTATION

Linked Data Reactor (LD-R) is implemented as an open source project which is available online at http://ld-r.org together with documentation and demos[5]. LD-R utilizes Facebook's ReactJS components, the Flux architecture, Yahoo!'s Fluxible framework for isomorphic Web applications (i.e. running the components code both on the server and the client) and the Semantic-UI framework for flexible UI themes. Figure 5 shows an screenshot of the LD-R environment to view and browse DBpedia scientists, and also to edit a sample dataset.

LD-R is currently used in the SMS[6] (Semantically Mapping Science) platform as the technical core within the RISIS.eu project to view, browse and edit data related to Science, Technology and Innovation (STI) studies.

4. RELATED WORK

Component-based software engineering (CBSE) has been an active research area since 1987 with numerous results published in the research literature [12]. Within the Semantic Web community, the main focus has been on enriching current service-oriented architectures (SOAs) with semantic formalisms and thereby providing Semantic Web services as reusable and scalable software components [13]. There have also been a few attempts to create Semantic Web Components by integrating existing Web-based components with

[5]http://demo.ld-r.org
[6] http://sms.risis.eu

Semantic Web technology [1, 4]. The Semantic Web Framework [2] was one of the first attempts in that direction to decompose the Linked Data Application (LDA) development requirements into an architecture of reusable software components. In most of the current full-stack LDA frameworks such as Callimachus[7] and LDIF[8] the focus is mainly on the backend side of LDAs and less attention is paid on how Linked Data is consumed by the end-user. There are also more flexible application frameworks such as OntoWiki [3] which provide UI widgets and extensions to expose Linked Data to non-Semantic Web end-users.

Besides these generic LDA frameworks, there are also approaches that focus on the development of user interfaces for LDAs. WYSIWYM (What You See Is What You Mean) [5] is a generic semantics-based UI model to allow integrated visualization, exploration and authoring of structured and unstructured data. Our proposed approach utilizes the WYSIWYM model for binding RDF-based data to viewer, editor and browser UIs. Uduvudu [7] is another approach to making an adaptive RDF-based UI engine to render LD. Instead of adopting Web components, Uduvudu employs a set of flexible UI templates that can be combined to create complex UIs. Even though the static templates do not provide enough interactions for editing and browsing data (in contrast to Web components), we believe that algorithms for automatic selection of templates employed in Uduvudu can be reused in the LD-R framework for automatic generation of configurations. Another similar approach is SemwidgJS [11] which brings a semantic Widget library for the rapid development of LDA UIs. SemwidgJS offers a simplified query language to allow the navigation of graph-based data by ordinary Web developers. The main difference between LD-R and SemwidgJS is that LD-R suggests a more interactive model which is not only for displaying LD but also for providing user adaptations based on the meaning of data. LD-Viewer [8] is another related Linked Data presentation framework particularly tailored for the presentation of DBpedia resources. In contrast to LD-R, LD-Viewer builds on top of the traditional MVC architecture and its extensions rely heavily on the knowledge of RDF which is a burden for developers unfamiliar with SW technologies. In addition to the LDA UI frameworks, there are several ad-hoc tools for Linked Data visualization and exploration such as Balloon Synopsis [9] and Sgvizler [10] which can be utilized as Web components within the LD-R framework.

In overall, what distinguishes LD-R from the existing frameworks and tools is its modern semantics-based isomorphic component-based architecture that combines reactive and reusable UIs with explicit semantics found in data to realize the idea of data-aware UIs.

5. CONCLUSIONS

This paper presented the Linked Data Reactor (LD-R) open-source software framework as a solution to implement data-aware user interfaces (UIs) – UIs that can understand users' data and can interact accordingly. The proposed semantically-enriched component-based architecture aims to provide better UI customization and personalization based on the meaning of data and the context of user.

We argue that bridging the gap between Semantic Web Technologies and Web Components worlds brings mutual benefits for both sides. On one hand, Semantic Web technologies provide support for richer component discovery, interoperability, integration, and adaptation on the Web. On the other, Web Components bring the advantages of UI adaptation, standardization, reusability, replaceability and encapsulation to current Semantic Web applications.

[7]http://callimachusproject.org/
[8]http://ldif.wbsg.de/

Acknowledgement

This work was supported by a grant from the European Union's 7th Framework Programme provided for the project RISIS (GA no. 313082).

6. REFERENCES

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

[2] R. G. Castro, A. G. Pérez, and M. n.-G. Óscar. The Semantic Web Framework: A Component-Based Framework for the Development of Semantic Web Applications. In *DEXA '08*, pages 185–189, Washington, DC, USA, 2008. IEEE Computer Society.

[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] O. Hartig, M. Kost, and J. C. Freytag. Designing component-based semantic web applications with DESWAP. In *Poster and Demonstration Session at the ISWC2008*, 2008.

[5] A. Khalili and S. Auer. Wysiwym – integrated visualization, exploration and authoring of semantically enriched un-structured content. *Semantic Web Journal*, 2014.

[6] A. Khalili, A. Loizou, and F. van Harmelen. Adaptive linked data-driven web components: Building flexible and reusable semantic web interfaces. In *ESWC2016*, pages 677–692, 2016.

[7] M. Luggen, A. Gschwend, A. Bernhard, and P. Cudre-Mauroux. Uduvudu: a graph-aware and adaptive ui engine for linked data. In C. Bizer, S. Auer, T. Berners-Lee, and T. Heath, editors, *Workshop on Linked Data on the Web (LDOW)*, number 1409 in CEUR Workshop Proceedings, Aachen, 2015.

[8] D. Lukovnikov, C. Stadler, and J. Lehmann. Ld viewer - linked data presentation framework. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 124–131, New York, NY, USA, 2014. ACM.

[9] K. Schlegel, T. Weißgerber, F. Stegmaier, M. Granitzer, and H. Kosch. Balloon synopsis: A jquery plugin to easily integrate the semantic web in a website. In *ISWC Developers Workshop*, pages 19–24, 2014.

[10] M. G. SkjÃęveland. Sgvizler: A javascript wrapper for easy visualization of sparql result sets. In *ESWC2012*, May 2012.

[11] 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*, pages 479–490, 2014.

[12] T. Vale, I. Crnkovic, E. S. de Almeida, P. A. da Mota Silveira Neto, Y. a Cerqueira Cavalcanti, and S. R. de Lemos Meira. Twenty-eight years of component-based software engineering. *Journal of Systems and Software*, 2015.

[13] H. H. Wang, N. Gibbins, T. Payne, A. Patelli, and Y. Wang. A survey of semantic web services formalisms. *Concurrency and Computation: Practice and Experience*, 27(15):4053–4072, 2015. 10.1002cpe.3481.