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 structured 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).

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
2.1 Value Components

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:

- **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.

2.3 Adaptation Engine

An adaptive UI 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

---

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

2 http://www.dbpedia-spotlight.org

3 http://gadm.org

4 http://en.wikipedia.org/wiki/Adaptive_user_interface
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 an 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 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](http://ld-r.org) together with documentation and demos. 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 a 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 SM-SI [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...
Semantic Web technology [1,2] 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 [4] and LDIF [2], 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 semantically-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 SemwidiJS [1] which brings a semantic Widget library for the rapid development of LDA UIs. SemwidiJS offers a simplified query language to allow the navigation of graph-based data by ordinary Web developers. The main difference between LD-R and SemwidiJS 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 and SemwidiJS that is 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 and SemwidiJS that is 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 and SemwidiJS that is 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.