2010 14th International Conference Information Visualisation

Semi-Automatic Generation of GUIs for RDF Browsing

Maria Teresa Pazienza, Noemi Scarpato, Armando Stellato
ART Research Group, Dept. of Computer Science, Systems and Production (DISP)
University of Rome, Tor Vergata
Via del Politecnico 1, 00133 Rome, Italy
{pazienza, scarpato, stellato}@ info.uniroma2.it

Abstract

In this paper we present an approach to automatic generation of GUI for browsing of RDF data based on observation of existing forms and their adaptation to available RDF graphs. The objective of such an approach is the rapid prototyping of forms and their associated queries by exploiting the vast amount examples that is already available from the Web, and trying to automate those steps requiring human intervention (form template extraction, query specification) for customizing found examples to developer's specific needs.

Keywords--- **RDF Browsing, Fresnel, GUI generation, Query Induction, SPARQL**.

1. Introduction

Interaction with RDF [1] data is a crucial aspect for the Semantic Web [2] vision, where independent machines will be able to mediate information between them and autonomously present them to the user in the most appropriate way.

RDF graph management deals with different task as storage, manipulation, presentation and visualisation of this kind of data. While the processes of interaction and manipulation of RDF data have reached a substantial maturity, those regarding visualization and interaction, though not in their infancy, are undergoing diverse interesting proposals [3] so that they may be well considered at the peak of their evolution.

The objective of this field of research is visualizing the RDF data in a visually comprehensible form and show its expressive power. This tasks consisting of specifying *which* information contained in an RDF graph should be presented and *how* this information should be presented. There are two major approaches to Semantic Web data visualization: adopting and applying existing Information Visualization solutions (as in [4]) or developing completely new techniques specifically tailored for the knowledge representation paradigms of the Semantic Web (like in [5]). The information that is visualized in forms originated from RDF data is a subgraph, which can be extrapolated by proper queries: selecting and composing these queries (for which a standard is available from early 2008, in the form of the SPARQL query language [6]) requires a combination of domain/technical expertise to be applied.

Another key point of Semantic Web Data Visualizations lies in the definition of the same graphic elements that are associated to domain information. They can be elected to expose a representation on their own, and Semantic extensions to RDF, such as RDFS/OWL or other standards such as SKOS may play a pivotal role in representing this information, through expressly dedicated ontologies of visualization, addressing merely graphical aspects such as graphical templates from which UI widgets are decorated, geometrical aspects (size, width, depth etc), or other aspects close to their binding with the source data (the order in which certain collections of resource are displayed, or the way they are clustered etc). It is very important to point up that the pairs of type: *<rdf_resource, representation>* are relevant information on their own, which can be exposed, collected and reused by generic RDF browsers and viewers according to the same paradigm which is proposed by the Semantic Web for resource shareability and reuse.

In this paper we take up the challenge of defining an architecture for semi-automatic:

- generation of graphical widgets for visualization of RDF data
- induction of SPARQL queries for their population

through selection of un-annotated samples (such as HTML tables, forms etc).

In the following section we describe related works on models for information visualization and the state of the art of RDF browser. In section 3 we sketch the basis for our proposal, in section 4 we provide details about the architecture and in section 5 we show how users will interact with this system. The conclusions analyzes the

1550-6037/10 $26.00 © 2010 IEEE
DOI 10.1109/IV.2010.47

possibilities of this technology and next directions for realization of this system.

2. Related Works

An important step towards data visualization has been taken by Fresnel [7]: an RDF vocabulary (actually, an ontology modelled after the OWL language) for RDF information visualization. Fresnel provides a generic way to define the presentation of context information and sharing this presentation knowledge between compliant generic RDF Browsers. Fresnel's two basic concepts are *lenses* and *formats*. Lenses define which properties of one or more RDF resources to display and their order of presentation. Formats determine how to render the resources, their properties and values.

Being the standard query language for RDF, SPARQL is obviously the first choice to be used inside Fresnel lenses to specify the graph patterns to be extracted and projected over the UI elements. Other possible choices are FSL (Fresnel Selector Language) or simple lists of RDF nodes which need to be shown. In general RDF browsers compliant with Fresnel require to be setup with one or more configuration files realized by experts of the considered domain. This file contains specifications for all the lenses that will be applied on data and their related formats.

The rationale behind Fresnel relies in applying to lenses and formats the same Semantic Web principles of openness, shareability and reuse that are applied to knowledge resources, by creating reusable pairs of ontologies/configuration files, which can be searched, browsed, filtered according to user specific needs, downloaded and finally applied to local browser.

According to Palmér et al. [8] it is possible to define an *Annotation Profile* that explicates the data that must be displayed and its template of representation.

The concept of Annotation Profile is derived from the concept of Application Profile [9]. Application profiles specify which metadata to use in a specific application while an Annotation Profile has the additional purpose of allowing automatic generation of user interfaces for the adopted metadata. An Annotation Profile is composed of a graph pattern model and of a form template model, the graph pattern model is responsible for capturing and creating subgraphs of considered RDF graphs, the form template model defines representation, order and grouping of each subgraph. This approach requires metadata and/or domain experts that define annotation profiles according to metadata vocabularies.

2.1. State of the art on RDF browsers

One of the main peculiarities of traditional web browsers is that they can work with any content, providing that it is specified according to some given standard they accept: RDF Browser do the same, by complying with Semantic Web paradigm of reusability and sharing of information.

There are many works on RDF Browsing in literature (many of which have lead to the realization of prototype tools). All of them, propose even really different strategies for browsing RDF data; some tools provide nested boxes layouts, as Haystack [10] and Tabulator [11], that is, recursively contained boxes of property value pairs.

Others combine link navigation with *facets*: facets are different dimensions, perspectives, of the underlying data. Often, the values of this dimension are hierarchically structured to represent relevant categorization of data driven by each perspective. Facets are used in many different RDF Browsers like Longwell [12] and /facets [13].

Another approach is graph representation: different tools have been developed to support the visualization of RDF graphs such as:

- RDFSViz [14], a visualization service for ontologies represented in RDF schema;
- OntoViz [15], a highly configurable ontology visualization tool integrated in Protégé ;
- IsaViz [5] a flexible tool for RDF graph visualization, with a number of functions for zooming, editing, searching and browsing the graph structure ;
- RDF Gravity [16] a tool for visualizing and navigating directed graphs built in RDF and OWL, with the possibility to zoom, search, filter out and visualize specific parts of RDF graphs;
- Cluster Map [3] a key component of the Spectacle system, used for the visualization of ontological data, with a very expressive and configurable interface;
- GVis [17] a general purpose, flexible and highly customizable graph visualization tool is, used in the context of the Hera project for visualizing large RDF graphs;
- Welkin [18]
- Semantic Turkey [19], a Knowledge Management and Acquisition tool, providing graph exploration of edited ontology.

RDF graphs may not be intuitive to understand, in particular when they are very large and the relationships between its concepts are numerous. Moreover, though quite self-explicative, also RDF needs some non intuitive constructs to represent its data, such an in the representation of n-ary relationships, which needs to reify relationships and constructs chains of blank nodes which need to be properly interpreted (and thus shown accordingly).

Last generation RDF Browsers use the Fresnel vocabulary to define patterns of representation, as in LENA [20] and in last versions of Longwell and IsaViz.

3. Overall Concept

The layer-cake of technologies and languages for information representation in traditional Web content identifies well separated levels of competence where

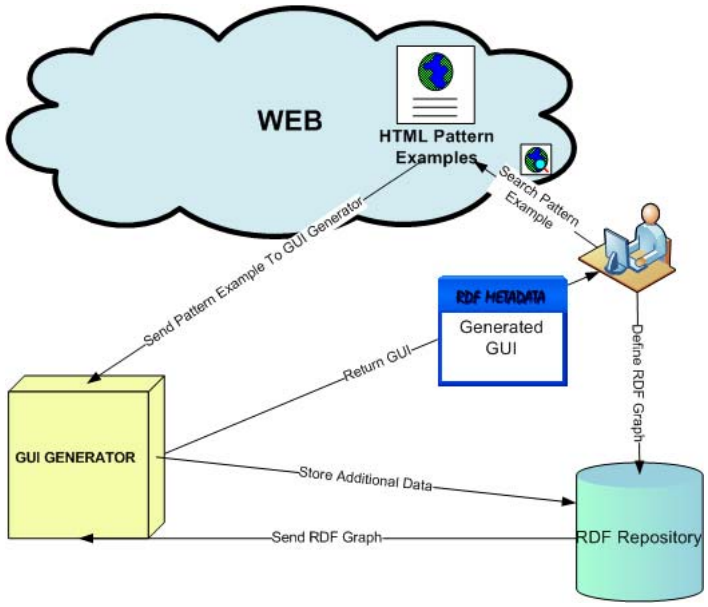

Figure. 1. System Design.

artistic work, content development and technological aspects may be assigned to the most appropriate figures.

Styles (e.g. CSS), content structure (HTML tags), embedded data (e.g. microfornats[1] and RDFa [21]) and server-side and client-side technologies for dynamic content publishing provide different levels of abstraction where all of the above figures find their role. This clear separation has led to highly specialized development tools allowing management of the aspects of interest for each figure, but also the proper abstraction from the other layers, and the simplification which is required for their competencies. In the same way, the RDF Browsers should provide a user interface that binds the graphical structure to metadata, and allow ontology experts, graphical artists and web/UI designers to cooperate under well defined interaction modalities. We want to go a step further towards this direction, by introducing a further level of abstraction which is provided by interdisciplinary work of domain experts, needing to provide the above developers with rapidly deployed mock-ups of desired interfaces, possibly already working at a basic level of detail (thus requiring some fine-tuning, which is requested to the developers). Our proposed approach is to devise a mechanism and a chain of processes (to identify a realizable architecture) that automatically generates the queries for extracting the desired subgraph, starting from available examples , and defines the right representation for the selected resources. By adopting Fresnel [7] vocabulary, we may then collect above information in a list of pairs: <*lenses* , *formats*> that will be used to configure the users interface and finally generate the GUI.

[1] http://microformats.org/

For each UI to be built for a given domain, there are surely lot of pages out there, on the (traditional) web, providing useful examples (tables, forms etc), which could be used. The contribution of these examples is two-fold:

1. They provide a sample of the graphical structures upon which the desired UI will be built
2. They tell how this structure needs to be populated

The second point is of particular importance: the found examples do not need to have been produced by the same dataset possessed by the user; they just need to contain data representing information originating from the same domain (or, at least, sharing a sensible overlap with it). This information can then be searched over the real dataset owned by the user, hoping that a good percentage of the data will be recognized upon it and that the system would thus be able to induct the queries needed to extract analogous data from the dataset.

The following use case describe a possible scenario of application of our induction mechanism: Mario has an OWL ontology describing the domain of football, and a lot of data for the past two seasons of the local football league modeled after this ontology. During his navigation on the web, Mario finds a table showing football players from the same league together with the teams in which they play, the number of goals marked in the current season and other interesting information. He thus uses its UI induction tool to realize an identical table, which is lively populated with data queried from its personal RDF dataset. He thus uses the tools by first highlighting the html source of the table, and then submitting its content to the system. The system then creates a Fresnel *format* abstraction for the given table (which is thus independent from the UI technology

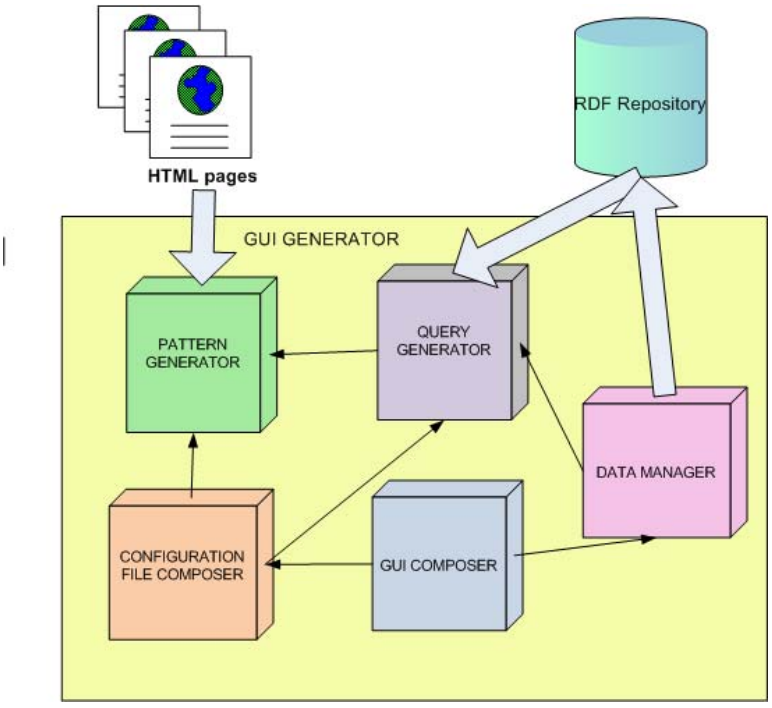

Figure. 2. Architecture.

which is being adopted) and then tries to induct the *most specific*[2] SPARQL query which will project the data over the table. Mario then submits the Fresnel output to its colleagues (web designer and RDF expert), which can then apply further refinement to the output. The UI extraction can be achieved by two alternative approaches: one is to apply wrapper induction [22] techniques, to extract the wrapping elements where the target information is contained, (so that they can eventually be found on other similar pages and extracted automatically), the other one is by simply recognizing predefined patterns (tables, lists etc) on the selected structure and project them over the Fresnel abstraction.

The induction of the queries is instead performed by applying ontology matching [23] techniques between the user RDF dataset and data contained in the example.

The results of those process have a double significance, on the one hand, the system has inducted the proper graph patterns to populate extracted form with live data from the dataset. On the other hand we can populate our RDF dataset with additional data extracted from similar pages. This is a collateral but interesting effect of applying our approach: the difference with respect to ordinary data mining (such as the above wrapper induction) is that the data template is not known a-priori, but is inducted from the available knowledge model.

[2] Since the found values are not necessarily coming from the same exact dataset, the inducted query may not contain all the constraints which select the exact subgraph fitting table values

At the end of the above process, the user has collected all the needed information to automatically generate a configuration file that contains lenses and formats.

Our approach can thus be reassumed in the following aspects:

- Configuration files are lively created when the user starts a learning process over data observed from a browsed example
- The patterns of representation are determine by the users
- The query identifying the interesting subgraphs is learnt by an automatic process;
- The information about the pattern of representation are stored and placed at user's disposal for future sessions of navigation (or to be exported for other interested users/developers).

3.1. Scenario

In figure 1 interaction between the identified components is being shown. These components and relevant objects are:

- A Semantic Repository containing RDF resources
- A GUI Generator that performed the automatic generation of user interface

and relevant objects are:

- HTML pages the content of which is being selected by the users
- The automatically generated UI

The RDF Repository contains one or more ontologies describing the domain of interest (the model) as well as the data provided by the user and/or collected retrieved during the processing of inputs. Also, data modeled for representation purposes and describing the created pairs *(rdf resource , representation)* through Fresnel *lens* and *formats*. Initially, if the RDF repository contains already enough amount of data, it can immediately be used as a seed to learn new UIs from available examples extracted from the Web, with no needed supervision. If it is empty and lot of similar examples are available from web pages, it can be automatically populated by semantically annotating even very few pages (as reported in [24], very few annotated examples are needed when applying wrapper induction techniques to very similar pages, which are usually produced by an original pattern populated by backed data). The GUI Generator receives as input html pages selected by the user (and highlighted parts of them), containing the formatted UI structure that the user wants to replicate, and the available RDF dataset. The GUI Generator analyzes the input and automatically generates a GUI mock-up with raw SPARQL queries for extracting plausible values. In the next section we describe in details the architecture of GUI Generator.

4. Architecture

In figure 2 the architecture of the UI Induction System is shown. The first component, named GUI Pattern Generator, performs the wrapper induction process, the goal of this process is to carry out the wrapper induction algorithm. Since the its first definition [22], many evolutions and implementation of different algorithms for wrapper induction have been realized, like in [25] or [24]; in particular there are algorithms performing wrapper induction on structured text (e.g. web pages) like [26]. As anticipated in section 3, the GUI Pattern Generator also applies heuristics-based processes oriented at recognizing predefined representation patterns and to extract data contained in it. For instance if the user submits part of a web page that contains a table, our algorithm extracts table elements such as *<table>*, *<tr>*, *<th>* tags and other information about content like number of columns and rows, furthermore it extract the content of columns head and the content of the rows of the table. The Query Generator receives both the data contained into the pattern recognized by the Pattern Generator component and the RDF graph provided by the user, and adopts ontology matching techniques to identify the best matching query. This query is expected to produce results which are *sound* upon a mere classification aspect (i.e. we expect to find resources constrained to the right headers in a table, such as instances of class *foaf:Person* under the header of the football players in previous example) though may not be properly constrained as to obtain the desired subgraph. In literature, there are many ontology matching techniques aiming at different purposes, in particular we are interested on methodologies supporting navigation on the semantic web, this techniques allowed to make tools like *PowerMagpie* [27] that is able to identify occurrence of instances of an ontology in a web page. The Configuration File Composer unifies inducted queries and related data to create the Fresnel *<lens, format>* pairs. Furthermore this component stores all pairs into the representation ontology that is put at hand of rdf resources furnished by the users into RDF Repository.

The GUI Composer component then generates a GUI (depending on the output technology which is selected for implementing the Fresnel abstraction layer) according to produced *<lens, format>* pairs. If are presents additional data, that are contained into example of representation provided by the users, the Data Manager component show them into generated GUI to the users that can validate this information. Finally the Data Manager component stores additional data into RDF repository.

5. User Interaction

Though partially automatic, our approach deserves a centric role for users, in that they provide the semantically annotated examples and where appropriate validate data retrieved from web pages.

The interaction of users with the system consists first of all in defining what is the RDF resources that they want to represent. Then they browse the web and search pages that have the same domain of the RDF repository. When the user meets a graphical pattern of representation for data he is interested in, then he selects it and asks the system to extract an UI widget and to induct a query for populating it with data from the available RDF resources, by comparing plausibility of results from the proposed query with respect to those presented in the selected examples. The system then starts the chain of processes and propose a widget to the user, with the option of first modifying and then saving the Fresnel format, then edit the query (e.g. to add more restrictions or simply change some of its characteristics) and finally save even the Fresnel lens. This last sequence of steps can be reiterated several times to refine the UI and change the associated query accordingly.

6. Conclusion and Future Works

In this paper we have defined an architecture to realize a GUI Generator able to analyze heterogeneous RDF resources and to generate for them dedicated UI from available samples.

The proposed approach can be implemented and integrated in very different scenarios, as an extension for Semantic Enhanced Web Browsers, RDF Browsers, Ontology Editors and Annotation Tools.

Users beneficiating of this application are both ontologists as well as domain experts.

We are currently implementing the GUI Pattern Generator and defining the query induction techniques.

Then we will implement the remaining components according to define architecture.

A future research direction for this kind of systems is in exploring the possibility of combining several configuration files to generate more complex GUIs, possibly specifying interrelationships (i.e. semantic constraints) between them. While this could simply be seen as a further refinement process resulting in more complex Fresnel configurations, we would stress the importance for the user of being able to specify compositional patterns for reusable atomic Fresnel units, in a sort of Semantic Mash-up. This would open up the way for reusable, shareable libraries of *active* UIs (i.e., bringing with them the information on how to populate them from available data), which should be easily searched (according to different perspectives, *what* they show, *how* they show it etc), accessed, imported (into heterogeneous Semantic UI developing environment) and composed according to user/developer needs, in the spirit of the Semantic Web vision.

7. References

[1] Graham Klyne, Jeremy J. Carroll, and Brian McBride. (2004, February) Resource Description Framework(RDF) :Concepts and Abstract Syntax,W3C Recommendation. [Online]. http://www.w3.org/TR/rdf-concepts/

[2] Tim Berners-Lee, James A. Hendler, and Ora Lassila, "The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities," *Scientific American*, vol. 279, no. 5, pp. 34-43, 2001.

[3] Vladimir Geroimenko and Chaomei Chen, *Visualizing the Semantic Web, XML-based Internet and Information Visualization*, Second Edition ed. London: Springer-Verlang, 2006.

[4] Tamara Munzner, "H3: Laying out large directed graphs in 3D hyperbolic space.," in *Proceedings of the 1997 IEEE Symposium on Information Visualization*, Phoenix, AZ, 1997, pp. 2-10.

[5] IsaViz Overview. [Online]. http://www.w3.org/2001/11/IsaViz/

[6] Eric Prud'hommeaux and Andy Seaborne. (2008, January) World Wide Web Consortium - Web Standards. [Online]. http://www.w3.org/TR/rdf-sparql-query/

[7] Emmanuel Pietriga, Christian Bizer, David Karger, and and Ryan Lee, "Fresnel - A Browser-Independent Presentation Vocabulary for RDF," in *5th International Semantic Web Conference*, Athens, GA, USA, 2006.

[8] Matthias Palmér, Fredrik Enoksson, Mikael Nilsson, and Ambjörn Naeve, "Annotation profiles: Configuring forms to edit RDF," in *International Conference on Dublin Core and Metadata Applications*, 2007.

[9] Thomas Baker, Makx Dekkers, Thomas Fischer, and Rachel Heery. (2000) Dublin Core application profile guidelines. [Online]. http://dublincore.org/usage/documents/profile-guidelines/

[10] Dennis Quan, David Huynh, and David R. Karger, "Haystack:A Platform for Authoring End User Semantic Web Applications.," in *2th International Semantic Web Conference*, Sanibel Island, Florida, USA, 2003.

[11] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, and Ruth Dhanaraj, "Tabulator: Exploring and Analyzing linked data on the Semantic Web.," in *3rd International Semantic Web User Interaction Workshop in International Semantic Web Conference*, Athens, Georgia, USA, 2006.

[12] (2003) SIMILE: Longwell RDF Browser. [Online]. http://simile.mit.edu/longwell/

[13] Jacco van Ossenbruggen and Lynda Hardman Michiel Hildebrand, "/facet: A Browser for Heterogeneous Semantic Web Repositories," in *ISWC*, 2006.

[14] Michael Sintek and Andreas Lauer. (2000) RDFSViz. [Online]. (http://www.dfki.uni-kl.de/frodo/RDFSViz/

[15] Michael Sintek. OntoViz. [Online]. http://protegewiki.stanford.edu/index.php/OntoViz

[16] Sunil Goyal and Rupert Westenthaler. RDF Gravity (RDF Graph Visualization Tool). [Online]. http://semweb.salzburgresearch.at/apps/rdf-gravity/

[17] Flavius Frasincar, Alexandru Telea, and Geert-JanHouben, "Adapting graph visualization techniques for the visualization of RDF data," in *Visualizing the Semantic Web: XML-based Internet and information visualization*. London: Springer-Verlang, 2006, ch. 9.

[18] welkin. [Online]. http://simile.mit.edu/welkin/

[19] Donato Griesi, Maria Teresa Pazienza, and Armando Stellato, "Semantic Turkey - a Semantic Bookmarking tool (System Description)," in *4th European Semantic Web Conference (ESWC 2007)*, Innsbruck, Austria, 2007, June 3-7.

[20] Jörg Koch and Thomas Franz, "LENA - Browsing RDF Data More Complex Than Foaf," in *7th International Semantic Web Conference*, Karlsruhe Germany, 2008.

[21] Ben Adida and Mark Birbeck. (2007, October) World Wide Web Consortium - Web Standards. [Online]. http://www.w3.org/TR/xhtml-rdfa-primer/

[22] Nicholas Kushmerick, "Wrapper induction for information extraction," 1997.

[23] Jerome Euzenat and Pavel Shvaiko, *Ontology Matching*. New York, USA: Springer-Verlag New York, Inc, 2007.

[24] Stephen Soderland, "Learning Information Extraction Rules for Semi-structured and Free text," *Machine Learning*, vol. 34, no. 1-3, pp. 233 - 272, February 1999.

[25] Ion Muslea, Steven Minton, and Craig A. Knoblock, "Hierarchical Wrapper Induction for Semistructured Informatin Sources," *Journal of Autonomous Agents and Multi-Agent Systems*, 2001.

[26] Costin Badica, Amelia Badica, and Elvira Popescu, "A New Path Generalization Algorithm for HTML Wrapper Induction," in *Advances in Web Intelligence and Data Mining* , 2006.

[27] Laurian Gridinoc, Marta Sabou, Mathieu d'Aquin, Martin Dzbor, and Enrico Motta, "Semantic Browsing with PowerMagpie ," in *The Semantic Web: Research and Applications*, 2008, pp. 802-806.