Chapter 5
A Semantics-Based, End-User-Centered Information Visualization Process for Semantic Web Data

Martin Voigt, Stefan Pietschmann, and Klaus Meißner

Abstract Understanding and interpreting Semantic Web data is almost impossible for novices as skills in Semantic Web technologies are required. Thus, Information Visualization (InfoVis) of this data has become a key enabler to address this problem. However, convenient solutions are missing as existing tools either do not support Semantic Web data or require users to have programming and visualization skills. In this chapter, we propose a novel approach towards a generic InfoVis workbench called VizBoard, which enables users to visualize arbitrary Semantic Web data without expert skills in Semantic Web technologies, programming, and visualization. More precisely, we define a semantics-based, user-centered InfoVis workflow and present a corresponding workbench architecture based on the mashup paradigm, which actively supports novices in gaining insights from Semantic Web data, thus proving the practicability and validity of our approach.

5.1 Introduction

With the advent of the Semantic Web technologies like RDF, RDFS, and OWL, more and more organizations publish their information as so-called *Linked Open Data* in the form of open semantic knowledge bases.[1] Consequently, there is an increasing need for tools to manage and process this rapidly-growing amount of data. One important aspect in this regard is how to enable end-users, i.e., knowledge workers, to analyze and gain insights from these data sets. Unfortunately, this task is mainly reserved to tech-savvy users (Dadzie and Rowe 2011). Here is why:

[1] As of February 2013, the *Data Hub* (http://thedatahub.org/) hosts about 5100 data sets from various domains.

M. Voigt (✉) · S. Pietschmann · K. Meißner
TU Dresden, Dresden, Germany
e-mail: martin.voigt@tu-dresden.de

S. Pietschmann
e-mail: stefan.pietschmann@tu-dresden.de

K. Meißner
e-mail: klaus.meissner@tu-dresden.de

Primarily, end-users lack an understanding of Semantic Web data, its syntax and structure. They may know spreadsheets and have an idea what the rows and columns mean. However, they do not (and need not) know about concepts like triples, multiple inheritance, or that properties are not hardly tied to classes. Hence, tools need to present Semantic Web data in a reasonable, understandable way.

Various RDF browsers (Dadzie and Rowe 2011) and ontology visualization methods have been proposed (Katifori et al. 2007). However, they are usually limited to graph- or list-based data representations and thus do not exploit capabilities of prevalent visual analytic systems, e.g., support for generic charts, multiple coordinated views, iterative mapping refinement, or the recommendation of visualizations. Even more important, they are tailored (and limited) to specific domains and data sets.

Unfortunately, well-established, generic InfoVis tools like Tableau[2] do not support Semantic Web data, and there is no sign this is going to change soon. While slowly, more promising concepts for generic RDF InfoVis are emerging like the SPARQL result set visualization from the *Data-Gov* project (Ding et al. 2010), they require users to employ expert knowledge in Semantic Web, programming and visualization.

Finally, even with proper Semantic Web InfoVis tools at hand, interpreting and finding the right visualization for a certain data set and goal is a challenging task for novices, because they lack the necessary visualization knowledge (Grammel et al. 2010). Knowledge-assisted visualization (Chen et al. 2009) tries to fill this gap by using formalized expert knowledge and reasoning. Despite innovation in this direction, existing solutions, such as (Kadlec et al. 2010; Wang et al. 2009), are domain-specific, self-contained, and not applicable for Semantic Web data. Furthermore, they do not recognize and incorporate context information of the used device, e.g., screen estate, and the user, e.g., explicit or implicit preferences, to present the data in a suitable manner.

In this chapter, we propose a novel concept for a generic, user-centered InfoVis workflow geared towards novices, which allows for the context-aware mapping of arbitrary data to appropriate visualization components. Further, we present the key challenges as well as our solutions for its application on Semantic Web data. Finally, we give an architectural overview of our InfoVis workbench VizBoard which implements our novel approach based on the mashup platform CRUISe (Pietschmann 2009) and, thus, allows for presenting any Semantic Web data in a dashboard-like, composite, and interactive visualization.

Our chapter is structured as follows: After giving a brief overview of related work in the next section, we introduce the foundations of our concept in Sect. 5.3: a context-aware application composition framework and a visualization knowledge-base. In Sect. 5.4 we define a novel, user-centered InfoVis workflow which employs shared semantics to assist the visualization process. Further, we highlight the challenges and solutions found while realizing this workflow for Semantic Web data.

[2]http://www.tableausoftware.com/.

Then, Sect. 5.5 presents a corresponding software architecture which realizes the process based on the mashup platform CRUISe to illustrate and evaluate its applicability. Finally, we discuss our findings and point out future work in Sect. 5.6.

5.2 Related Work

Both the Semantic Web and InfoVis have received lots of attention by the research community in the recent years. Thus, we need to analyze existing concepts with respect to the goals and challenges lined out in the previous section. First, we give an overview of how knowledge models can assist the visualization process in general. Thereafter, we analyze existing generic approaches for the visualization of Semantic Web data.

5.2.1 Understanding and Supporting the Visualization Process

As mentioned before, the vision of a semantics-based InfoVis for novices requires both a formal knowledge model and a structured process which defines how to bridge the gap from raw data to an appropriate graphical representation. To this end, various visualization-specific process models and InfoVis concepts addressing novices have been proposed.

The pipeline model is commonly used to describe visualization as a process. In its elementary version it defines a sequence in which raw data is filtered and enriched, mapped to an abstract visualization specification, and finally rendered to a displayable image (Haber and McNabb 1990). This model has been successively enhanced, e.g., to include the user and his tasks (Card et al. 1999) or to allow for the coordination of independent views (Boukhelifa et al. 2003). In contrast to our work, the pipeline model focuses firstly on system-side functionalities and not on the (lay-)user in his struggle to gain insights from his data. Further, it does not employ formalized knowledge to represent which graphic representation is the best within a specific context.

Within the area of knowledge-assisted visualization, several authors have proposed ways to support the visualization process using knowledge models. Wang et al. (2009) describe how knowledge "moves" through the visualization process in a number of conversion steps, e.g., to externalize tacit user knowledge to explicit system knowledge. Yet, information on how to employ these steps in generic InfoVis systems to assist users in visualizing (Semantic Web) data, is missing. Chen et al. (2009) sketch a high-level knowledge-based infrastructure in parallel to the visualization system, which extracts information from data and uses it together with predefined expert knowledge to adapt the visualization process. Despite the similar goals, users' interaction steps and the integration of the formal knowledge in every stage of the InfoVis process is missing.

Both the pipeline model and knowledge-assisted visualization are primarily focusing on how a system can create appropriate visualizations. An additional, orthogonal aspect we consider important in our work is active user support. The first notable guidelines in this direction were given by Heer et al. (2008). They include easy data input, user assistance in selecting graphical representations, and the use of default mappings from data to visual variables. These principles have been underpinned by a recent user study (Grammel et al. 2010), wherein the authors suggest some additional guidelines and requirements, such as (semantics-based) search facilities to narrow the data set, adaptation to the iterative nature of the visualization process, and support for partial and uncertain input specifications of novices. Finally, Shneiderman's mantra (Shneiderman 1996) defines the most fundamental design guideline for all interactive systems addressing information search: "Overview first, zoom and filter, then details-on-demand". This is especially true for novices, who need a lightweight overview of the Semantic Web data before they dive into details in an iterative way afterwards.

In summary, previous work shares our goal of actively supporting novices during the InfoVis process by providing valuable advices. However, only Grammel et al. emphasize the power of semantics to support novices.

5.2.2 Information Visualization of Semantic Web Data

With the growing amount of Semantic Web data sets, more and more methods (Katifori et al. 2007) and tools (Dadzie and Rowe 2011) for their visualization have been proposed. Mostly, they focus on text- or graph-based visualization and are tailored towards special purposes and data sets. In the following, we focus on the few very generic InfoVis approaches.

An increasing number of US governmental data is made accessible in RDF (Ding et al. 2010) by the Open Government Directive. Tutorials on their visualization using popular APIs and widget libraries are published[3] which imply, that every user has the freedom to build his or her InfoVis of choice. Unfortunately, these tutorials—including a proxy for data transformation—are little help for novices, as Semantic Web, programming, and visualization skills are needed for their use.

Alternatively, the UISPIN framework provides means to describe user interfaces for rendering Semantic Web data. This includes a chart library[4] with various widgets to visualize Semantic Web data. The library can be embedded in Semantic Web tools, as it is the case for the TopBraid Composer.[5] Thereby, users can include charts without programming skills, but still need to define SPARQL queries for the data to be visualized. As further assistance, e.g., recommendation of suitable widgets, is

[3]http://data-gov.tw.rpi.edu/wiki/How_to_use_Google_Visualization_API.

[4]http://uispin.org/charts.html.

[5]http://www.topbraidcomposer.com.

missing, users must know, which visualization to choose and how to define queries in SPARQL.

A solution to one of these problems is given by Leida et al. (2010), who annotate SPARQL queries with a shared vocabulary of visualization-specific concepts to (semi-)automatically map RDF data to graphic representations. Since this promising approach focuses on the mapping only, a concrete semantic model for defining visualization-specific knowledge is missing as well as its integration in an overall, (lay-)user-centered InfoVis workflow.

Finally, Mazumdar et al. (2012) propose the *.view.* framework which employs the dashboard metaphor to visualize Semantic Web data with well-know charts in multiple views. We are also developing an interactive system to provide composite visualization of any RDF data but our approach is more sophisticated as we employ semantic models to allow for a context-aware, automatic mapping of data to the widgets without the need to manually define any configuration files for the data set. Furthermore, we provide a user-centred workflow comprising a data filtering and widget selection geared towards novices.

All in all, current solutions from this field solely focus on the visualization of SPARQL query results. Their common limitation on SELECT statements implies, that graph-based visualizations are mostly excluded, even though these are better suited and commonly used for Semantic Web data. With these concepts we share the idea of combining arbitrary data sources with existing, web-based widgets from different libraries, following the mashup paradigm. However, and most importantly, prevalent solutions do not support novices adequately.

Before we present our concepts of a user-centered visualization process, the next section provides details about the conceptual and practical basis we are building on.

5.3 Conceptual Foundation

As can be seen from the discussion so far, realizing a context-aware InfoVis workflow is far from trivial, since a broad number of challenges has to be addressed. To this end, our solutions and the corresponding tooling are built on top of existing concepts and practical results from other research projects.

Most importantly, we use the concept of *universal* application composition which allows us to freely combine two types of building blocks: (semantic) data sets and generic visualization components. This composition is supported by visualization knowledge formalized as an ontology. In the following, we present some insights on these foundations.

5.3.1 Universal Context-Aware Mashup Composition

In our work, we build on the results from the CRUISe project (Pietschmann 2009), which provides a conceptual foundation as well as an ecosystem for the dynamic,

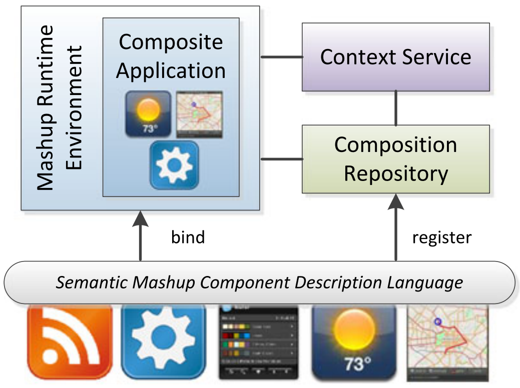

Fig. 5.1 Architectural overview of the CRUISe ecosystem

context-aware composition of web applications from distributed building blocks. The following paragraphs provide a brief overview of the corresponding concepts and infrastructure parts illustrated in Fig. 5.1.

The idea of universal composition implies a uniform component model, to which all parts of an application adhere. Such components are black-box pieces of independent software that provide a dedicated functionality. It is important to note, that this explicitly includes user interface, i.e., visualization components to be reused in different contexts.

In our conceptual space, components are characterized by three abstractions, namely *Property*, *Event*, and *Operation*. The set of properties resembles the component state and allows for its configuration. Whenever the internal state changes, events are issued to inform the runtime system and other components. Finally, state changes, calculations and other arbitrary functionality of a component can be triggered by invoking its operations with the help of events. Events and operations may themselves contain semantically typed *parameters*, thereby realizing the data flow between components.

Using the *Semantic Mashup Component Description Language* (SMCDL), components are described in a platform-independent, declarative way—comparable to WSDL for the description of web services. SMCDL is used to specify the above-mentioned interface parts as well as non-functional properties and information on how concrete implementations are bound to the abstract interface at runtime.

End-user-oriented authoring tools are employed to create interactive applications from these components, e.g., including search and recommendation features. Those applications can be expressed formally as instances of the Mashup Composition Model (MCM) (Pietschmann et al. 2010)—a description of the components, the data and control flow, the visual layout and the adaptive behavior of a composition on a platform-independent level.

For the visualization of Semantic Web data, our goal is to create a step-by-step InfoVis workflow which semi-automatically binds generic visualization components to given data providers, e.g., by finding and recommending suitable components with respect to a given context, resulting in a composite application.

For the interpretation and execution of universal compositions, CRUISe has come up with a reference architecture of a *Mashup Runtime Environment* (MRE) and the corresponding infrastructure. During the model interpretation, a MRE requests components from a given Component Repository (CoRe). The latter always returns those component instances, which fit the application requirements and context best. In this discovery process, both implicit and explicit rules are used, e.g., to consider the technological compatibility (implicit) or user preferences (explicit). There are more services involved in the composition, but those are the main ones involved in our approach.

This integration process and composition infrastructure form the basis for our InfoVis workflow, as it allows to include visualization knowledge in the dynamic, context-aware composition of applications. To realize this vision, a formalized representation of this knowledge is required, though. Therefore, the next section introduces a modular visualization ontology, which does just that.

5.3.2 Formalizing Visualization Knowledge

The fundamental problem of InfoVis, regardless if done manually or automatically, is to find an appropriate mapping between data and visual attributes. Therefore, visualization knowledge is required. Tools like Tableau already provide limited support for novices, who lack this kind of knowledge. However, they do not cover the complete parameter space, e.g., including the used device or users' preferences. As mentioned in Sect. 5.2, few approaches facilitate semantic technologies to assist the visualization process, yet a generic, formal, and freely distributed knowledge model is still missing.

For this reason, we developed the modular visualization ontology (VISO, cf. Fig. 5.2-1) (Voigt and Polowinski 2011). It provides a well-documented vocabulary of concepts and relations to formally describe data, graphics, human activity, as well as the user and system context. Since we focused on the first two modules, we re-used existing and well-established ontologies whenever possible, such as the DEMISA task ontology (Tietz et al. 2011). Based on the defined entities, we also modeled factual expert knowledge (cf. Fig. 5.2-2), e.g., that using *position* instead of *color coding* is more suitable to visualize *quantitative* data, which is used to rank different mapping alternatives. Equally, users' input data (cf. Fig. 5.2-3) can be annotated with visualization semantics, e.g., an RDF property *price* may have a *quantitative* scale of measurement and an assigned domain *UnitPriceSpecification* of the GoodRelations ontology.[6]

With the help of VISO, user interface components of the CRUISe ecosystem can be described (cf. Fig. 5.2-4) with regard to visualization specific aspects. Therefore, the domain-independent SMCDL is extended to link to VISO concepts and properties. We annotate the data structures of the component interface—in Operations,

[6]GoodRelations ontology: http://purl.org/goodrelations/v1.

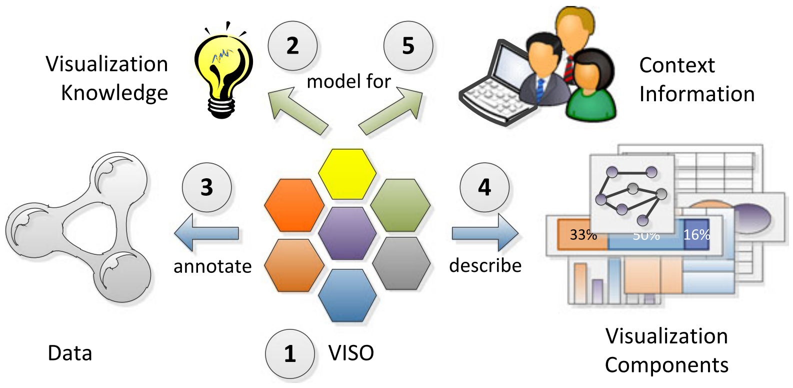

Fig. 5.2 Generic visualization ontology (VISO) as conceptual foundation of our visualization workflow (reprinted from Voigt et al. 2012d)

Events, and Properties—the kind of graphic representation used (map, scatter plot), the visual complexity (high, low), or the interaction potential (zoom, filter). Finally, the user and system contexts (cf. Fig. 5.2-5) are represented based on CRUISe's context service (cf. Fig. 5.1), e.g., in terms of *preferences* and user *skills*, the *display size* or the available *software* infrastructure.

All in all, by using VISO as a common vocabulary, all stakeholders of an InfoVis process, including contextual information, are combined in one knowledge base, thereby facilitating the context-aware recommendation of visualization components. In the following section, we present the steps of this semantics-driven InfoVis workflow in detail.

5.4 Context-Aware Information Visualization Workflow for Semantic Web Data

To address the problems lined out in Sect. 5.1, we propose a novel interactive, user-driven InfoVis workflow (cf. Fig. 5.3) which builds on the common semantic vocabulary provided by VISO and some insights retrieved from related work discussed in Sect. 5.2. The workflow can be applied to arbitrary data models, however, the following discussion specifically focuses on the visualization of RDF data and the corresponding challenges.

The workflow design is inspired by the way (lay-)users naturally interact when analyzing data. It consists of five stages users needs to pass: choosing or uploading a data set (cf. Fig. 5.3-1), getting an overview of the data and choosing a subset (cf. Fig. 5.3-3), selecting relevant data variables and suitable visualization components (cf. Fig. 5.3-5), configuring them (cf. Fig. 5.3-7) and, finally, interacting with the rendered data to gain the desired insights (cf. Fig. 5.3-9). Due to the interactive nature of the visualization process, users can sequentially pass through, but may

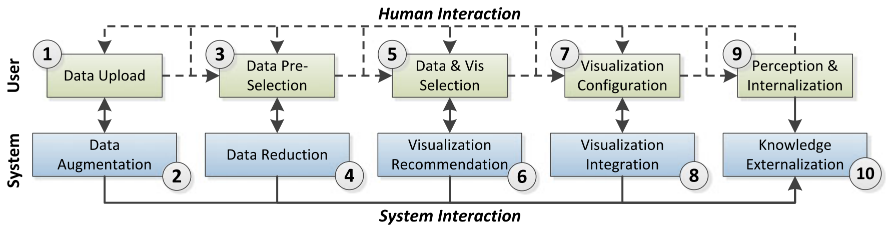
Human Interaction
User
System
1 Data Upload
3 Data Pre-Selection
5 Data & Vis Selection
7 Visualization Configuration
9 Perception & Internalization
Data Augmentation 2
Data Reduction 4
Visualization Recommendation 6
Visualization Integration 8
Knowledge Externalization 10
System Interaction

Fig. 5.3 Overview of the semantics-based visualization workflow (reprinted from Voigt et al. 2012d)

also move backwards. For instance, the configuration step can be skipped by using default mappings. Furthermore, users may choose to search and integrate multiple, alternative visualizations to benefit from multiple coordinated views of their data after completing the workflow.

This user-driven process is supported by five system-side functionalities which make use of the VISO (the lower rectangles in Fig. 5.3). Elementary functionalities like storing, querying, and supplying the data, graphic representations or knowledge are omitted from the figure for the purpose of simplification. In the following, we discuss each step of the workflow, point to major requirements and obstacles to realize them, and—as far as we already solved them—present our solutions.

5.4.1 Data Upload and Augmentation

The starting point of every visualization process is the provision of the data set, i.e., *raw data* (Card et al. 1999) (cf. Fig. 5.3-1). This data first needs to be transformed into a suitable format for the remaining process steps. After this, it must to be augmented (cf. Fig. 5.3-2) with visualization-specific knowledge, e.g., the kind of *scale of measurement* (nominal, ordinal, or quantitative), using the VISO vocabulary. This (semi-)automatic augmentation is the foundation for nearly all of the following system-side tasks, like the recommendation of appropriate visualization components or their coordination support.

The **Data Upload** is the most trivial part of the complete workflow. It requires the user to select an RDF or OWL file, a URI of a data dump or a Web service API and to submit it to the visualization system. It is also possible to support other data formats, like tabular (spreadsheets, etc.) or relational data sets (MySQL database) through a transformation step, as indicated in Fig. 5.4-2. In these cases, the data needs to be transformed into RDF triples using corresponding APIs. Especially for the mapping from relational databases to RDF a broad range of tools is already available (Sahoo et al. 2009).

Data Augmentation is split in two parts: First, evident visualization knowledge about the data is reasoned using information-retrieval techniques (cf. Fig. 5.4-3). Second, the data is augmented with this semantics using the VISO vocabulary

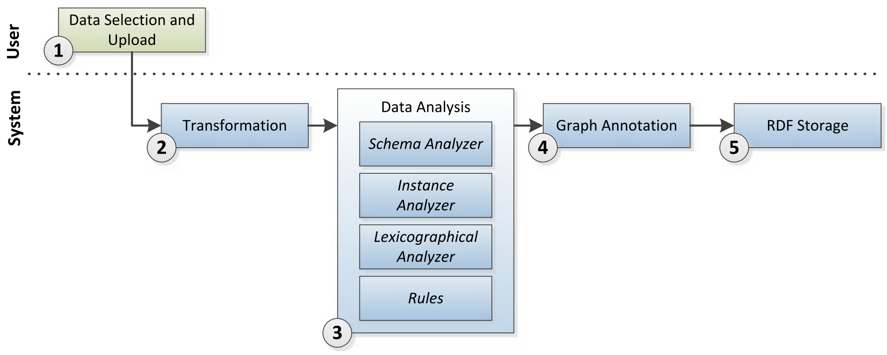

Fig. 5.4 Data upload and augmentation in more detail

(cf. Fig. 5.4-4). Here, the benefits of the Semantic Web come in handy, as the data can be easily linked to other concepts.

In this augmentation step, four distinct analyzers can be employed: (1) a *schema analyzer* which extracts information about simple data types if they are not explicitly provided; (2) an *instance analyzer* which calculates metrics like the number of distinct instances; (3) a *lexicographical analyzer* to identify more generic concepts as well as categories from DBPedia[7] with help of the WordNet[8] knowledge base to provide additional information to support the data to visualization mapping step; (4) a *rule engine* can be used to add custom relationships in a flexible manner. A common problem is the automatic identification of the *Scale of Measurement* of a property according to its basic data type, instances, and already identified domain concepts. A typical example would be to identify that a property called "school grade" has an ordinal scale instead of a nominal.

In the end, the annotated RDF graph needs to be stored within a homogeneous data layer—an RDF triple store—to allow for system-wide uniform data access in the following workflow steps (cf. Fig. 5.4-5). As a side note, we suggest to include a manual step in order to let an expert check and edit the automatically generated annotations and the declarative rules.

5.4.2 Data Pre-Selection and Reduction

One of the key problems of a generic approach to visualize Semantic Web data is the size of the data sets. In contrast to the findings of Sicilia et al. (2012) that most OWL ontologies are small and flat, there exist quite a number of huge OWL ontologies, especially in the public and medical sectors, e.g., the NCI thesaurus.[9] Moreover,

[7]DBPedia: http://dbpedia.org/About.

[8]WordNet: http://wordnet.princeton.edu/.

[9]http://ncicb.nci.nih.gov/download/evsportal.jsp.

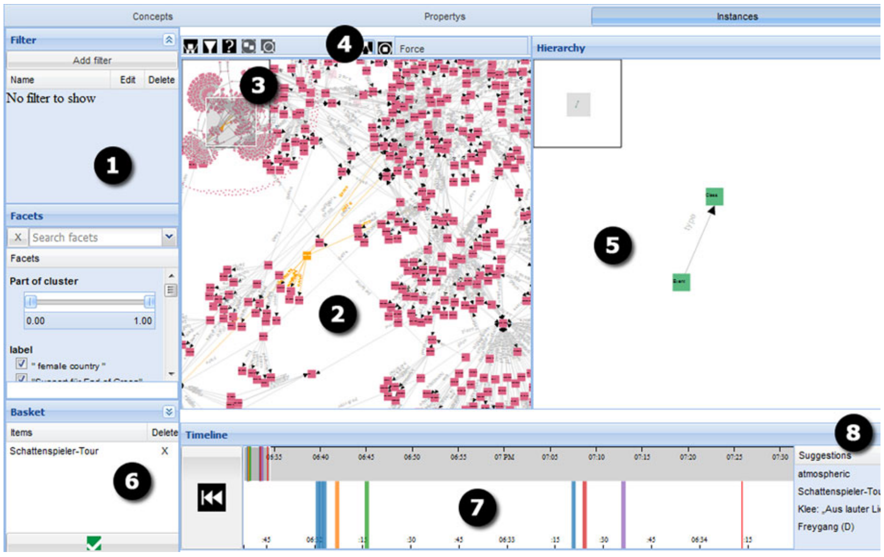

Fig. 5.5 Screenshot of our first prototype of the pre-selection

OWL has not yet "arrived" in the Linked Open Data (LOD) cloud (Glimm et al. 2012), which heavily relies on RDF and RDFS sets. Currently, it comprises 295 sets with approximately 32 billion triples, which implies an average 107 million triples per data set.[10]

In order to handle this amount of data, two challenges have to be addressed: First, the data sets must be visualized in an understandable, interactive manner with the goal to select classes, properties, or instances for more in-depth informations visualizations (cf. Fig. 5.3-3). Second, techniques are required to reduce the data sets to the relevant entities and point to interesting areas for the user respectively (cf. Fig. 5.3-4). In the following, we present our solutions to these challenges.

Based on Shneiderman's mantra, the purpose of the **Data Pre-Selection** (cf. Fig. 5.3-3) is to give users a high-level view of a data structure. Through interactions like zooming, panning, searching, or filtering he is able to find interesting subsets of the data which are selected for an in-depth Information Visualization through suitable components afterwards. For our InfoVis workflow this means to provide novices with intelligent visualizations and convenient metaphors to interact with data sets of more than a million entities.

A first prototype of a corresponding user interface is shown in Fig. 5.5. Its development is based on best practices from related tools, e.g., the TopBraid Composer[11] as well informal user studies with software prototypes. The frontend comprises the

[10]Information based on the State of the LOD Cloud report from October 2011, http://www4.wiwiss.fu-berlin.de/lodcloud/state/.

[11]TopBraid Composer: http://www.topquadrant.com/products/TB_Composer.html.

following parts: At the left, we offer various options to filter the data (1) by terms and facets, which are discussed below. The main view (2) shows all resources of the selected type (classes, properties, or instances). Within this view, users may zoom and pan while always having an overview of the dataset in (3). Further, the prototype offers methods for clustering, key concepts extraction, and path finding as well as different graph layouts (4). On the right (5), users may see and traverse the hierarchy of the resources selected in the main view. Of course, the size of both views can be adapted on-demand. Our solution also offers a "basket" (6), which allows to bookmark and collect resources of interest for later investigation or visualization. At the bottom (7) a time line shows breakpoints of user interactions in different colors, e.g., setting up a filter or zooming. Clicking them allows users to undo their interactions up to this task. Finally, the UI suggests interesting resources (8) depending on user selections, which are calculated using the pivoting algorithm sketched below.

All in all, the user interface for the Data Pre-Selection is functionally rich and allows for a versatile navigation and reduction of the dataset. Unfortunately, preliminary user tests show that lay-user are still overburdened to some extend. Hence, we are going to conduct a broader user study to identify the right balance between functionality and user satisfaction.

To assist the Data Pre-Selection in the frontend, a **Data Reduction** (cf. Fig. 5.3-4) must take place. Therefore, we suggest to use different data mining strategies as proposed in Fayyad et al. (1996): classification, clustering, summarization, or link analysis. Unfortunately, many of those techniques are commonly geared towards tabular data or relational databases. Thus, an adaptation is required which necessitates a distinction of the different ingredients of an ontology, namely classes, (object and data) properties, and instances. Currently, our conceptual workflow includes the following techniques in combination to allow for differentiated data reduction.

Faceted-Based Filtering Based on different kinds of metrics, facets and facet values can be created to allow for a target-oriented data filtering. The metrics calculation depends on the resource type. For classes we suggest to use topological characteristics like the number of subclasses, the number of instances, and the betweenness (Brandes 2001). Properties can be filtered using their domain, range, and hierarchy. Finally, instances may be distinguished by their class membership.

Clustering A number of different clustering algorithms like the wide-spread k-Means algorithm allow to find and summarize similar entities. The metrics mentioned above can also be applied as distance functions for them to cluster classes, properties, and instances.

Path Finding Another concept we employ for reducing the data set is path finding. Here, the idea is to calculate the shortest path(s) between two or more classes or instances of interest, and to filter out all resource outside these paths.

Key Concept Extraction Furthermore, we include the *key concept extraction* (Peroni et al. 2008) approach to identify and highlight the most relevant resources within a data set. Therefore, it makes use of insights from cognitive science, network algorithms, and lexicographical statistics. However, this solution is only applicable for classes.

Pivoting Another way to provide only a small subset of the data and to extend it on-demand is called pivoting (Popov et al. 2011). To calculate potentially interesting items we apply different metrics, e.g., the topological similarity for classes and properties or the semantic similarity for instance data.

Association Rule Mining Finally, the tracking of navigation trails within the data set in the Data Pre-Selection are analyzed using association rule mining techniques. Its results allows to highlight interesting resources or reduce the cumbersome information overload.

5.4.3 Interactive Data and Visualization Selection

After a user has narrowed down his data set to a region of interest, the following **Selection** step (cf. Fig. 5.3-5) covers the exploration and selection of interesting data variables and suitable visualization components to represent them. This is especially challenging for end-users, as their lack of InfoVis knowledge often leads to unsatisfying visualizations results (Grammel et al. 2010). In order to assist them, the workflow must include support mechanisms, e.g., suggesting appropriate graphical representations based on the selected data attributes and visualization characteristics. While the latter recommendation algorithm is explained in Sect. 5.4, the following paragraphs focus on the user interface and interaction.

For the design of a suitable search interface in this context it is fundamental to decide between querying and browsing. For the user it is less mental work to scan and choose from a list of entities than to think about appropriate query terms to describe his information need (Hearst 2009). Thus, with respect to our target group of novices, we suggest to use a browsing approach—in particular the interactive *faceted browsing* paradigm. Thereby, empty result sets can be avoided and users gain immediate feedback and can refine their queries iteratively. However, in our research we also faced some problems with using faceted browsing for data and visualization selection. Most importantly, users need to assign priorities to facets within a search query. Thus, we extended the paradigm to *weighted faceted browsing* introduced in Voigt et al. (2012b). In the following, be briefly describe these novel concepts, which address the definition of search criteria, the result ranking, and the corresponding, intuitive user interface.

First of all, we distinguish between mandatory and optional search criteria. To narrow the results, a facet value needs to be added to the mandatory set where all criteria are linked conjunctively—the standard behavior of a faceted browser. In contrast, optional facets are combined disjunctively within a dedicated set and thus do not constrain but rank the results. That way, the more optional criteria an item satisfies the higher it is ranked. If multiple items meet the same number of optional criteria, their ranking is the same. However, every criterion may be given an explicit weight (between 1 and 100), which directly influences its ranking. Zero value is neglected, as this means that facet can be omitted.

The input for calculating the overall result set is a query set of mandatory and optional criteria. While the mandatory part simply constraints the set by removing

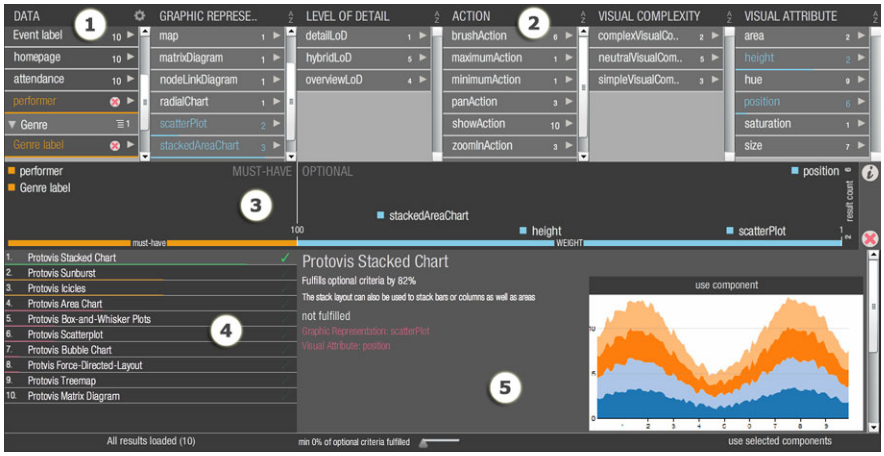

Fig. 5.6 Screenshot of our weighted faceted browsing prototype (reprinted from Voigt et al. 2012b)

all items not supporting a chosen facet value, the relevance ranking using multiple optional facets is more complicated. To solve the multi-criteria optimization, we combine the *weighted sum model* with *lexicographic ordering* in an iterative way to interpret the criteria. If some elements still have the same weight we order them alphabetically.

Of course, these theoretical concepts of weighted faceted browsing need not be visible to the lay-user. Instead, we have designed an intuitive user interface which consequently builds on the principles of existing facet browsers. The running prototype is shown in Fig. 5.6. As can be seen, the view is split into three main areas: facet widgets at the top (1), (2), the query visualization—called *querycloud*—in the middle (3), and the results view at the bottom (4), (5). To search for a visualization component, the user simply needs to drag a desired facet value—a RDF resource to visualize (1) or a visualization characteristic (2)—and drop it at a desired spot in the querycloud which is split into a mandatory and (weighted) optional area. The result set visualization (4) updates subsequently. By selecting an item from the result list, detailed information are displayed in (5).

We conducted a preliminary user study to test our hypotheses and the practicability of weighted faceted browsing for visualization selection. After an introduction, users had to handle five basic and five advanced search tasks to find appropriate visualization for given data sets. To our delight, the subjects answered all questions correctly. They generally enjoyed the intuitive approach, in particular of the querycloud. Whereas the basic tasks were solved without any help, we needed to give some assistance at solving the advanced issues. This was mostly caused by the missing understanding of the data set and the metadata of the visualization components within the facet widgets. An exemplary questions was: “What does *neutral visual complexity* mean?”. Thus, providing additional information on facet values should prove beneficial.

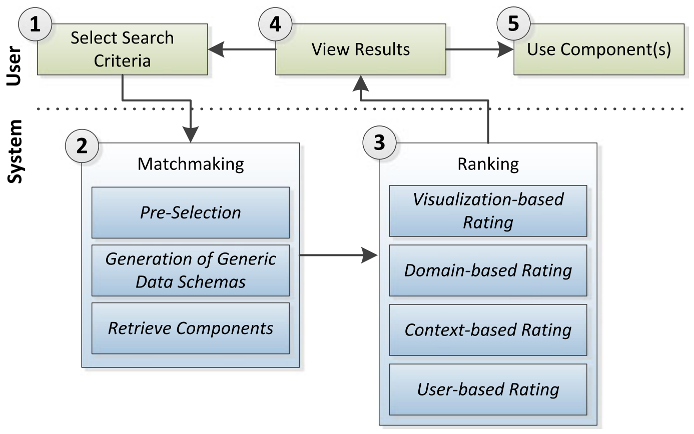

Fig. 5.7 Overview of our visualization recommendation process

5.4.4 Context-Aware Recommendation of Visualization Components

With respect to the knowledge and experience of novices, the exploration and selection of data variables and visualization facets as described in the previous section should be actively supported with **Visualization Recommendation** techniques (cf. Fig. 5.3-6). Of course, this functionality constitutes the heart of every visualization process and it comes with a number of challenges: First, an algorithm needs to discover appropriate visualization components based on the selected Semantic Web data using the aforementioned semantic annotations (cf. Sect. 5.3). Second, the algorithm needs to rank every identified component due to its applicability within the current context. In the following, we summarize the main concepts of our recommendation algorithm which is described in more detail in Voigt et al. (2012a).

Figure 5.7 gives an overview of our recommendation process. It starts with the selection of search criteria (1) which is realized by the weighted faceted browser. Based on the selection, the matchmaking process (2) starts with a pre-selection step. Thereby, the amount of components is reduced by matching the visualization-specific criteria, e.g., the kind of representation, the level of detail, or the interaction potential needed. Afterwards, a generic data schema is generated by mapping the data structure selected by the user to a generic one based on VISO. This schema then forms the basis for the retrieval of appropriate visualization components.

Subsequently, the list of suitable components is ranked (3) making use of the semantic annotations and VISO rules introduced earlier. The ranking step includes four criteria: First, the appropriateness is calculated with respect to visualization-specific knowledge, e.g., the visual encodings for quantitative data (Cleveland and McGill 1984). Second, the assigned domain concepts or categories of the data variables and the visualization component are employed. For each combination of those, the semantic similarity is calculated. Third, contextual knowledge is included. Thus,

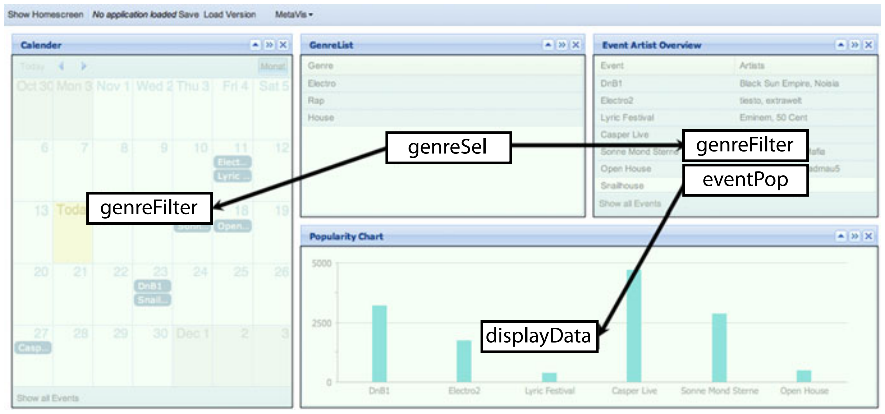

Fig. 5.8 Screenshot of the meta-visualization which allows to show and edit the coordination between the integrated components

the context model is queried for user or device characteristics, such as the screen real estate, in order to identify the best fitting visualization. Last but not least, we consider user-based ratings of existing components. The collection of this information is further described in Sect. 5.4.

Finally, combined rating for each component allows to establish a ranking order. This sorted list is presented to the user, who may either adjust his search criteria (1) or select one or more components to visualize the selected data with (5).

5.4.5 Visualization Integration and Configuration

Having selected one or more components to visualize his Semantic Web data, the work of the user is done. For the underlying system it means, that all those components need to be loaded and instantiated, "bound" to the selected data, configured with respect to the users' needs (cf. Fig. 5.3-7), and integrated with each other resulting in a homogeneous user interface (cf. Fig. 5.3-8).

For the integration, we heavily build on the concepts and infrastructure of CRUISe (cf. Sect. 5.3). It already provides means to load the selected, possibly distributed components and to manage their life cycle including instantiation and configuration. In a next step, we establish the "binding" to the selected Semantic Web data. The corresponding data and mapping information directly result from the previous workflow steps. By choosing several visualization recommendations iteratively in the previous steps, users are (implicitly) building *multiple coordinated views* of their data, which are finally presented interactively using CRUISe's runtime platform MRE.

Figure 5.8 shows such a user interface, overlayed with a meta-visualization. As can be seen, the data set—in this case concert data—is shown with respect to four different aspects (time, genre, artist, popularity) and corresponding visualizations (calendar, lists, and a bar chart).

The meta-visualization in Fig. 5.8 is one way to let novices establish (or edit existing) coordination links. The latter allow for the integration of visualization components on the data level by connecting their data variables for synchronization or filtering. This way, selected data in one view, e.g., the genre in this example, can be highlighted or act as a filter in another one. By establishing such links, users intuitively create *coordinated multiple views* on their data sets from which they may gain a better understanding of the displayed information.

Further mechanisms for user-driven adaptation may be provided, such as component configuration, which highly depends on their implementation. Typical configuration parameters include color schemes and filters. If multiple mappings between the underlying data and the component interface are possible, the user may as well adapt them, e.g., to switch the data mapping between the axes of a scatter plot. More sophisticated adaptation is possible as well, such as component exchange or layout changes. These mechanisms are provided by the underlying platform—in our case CRUISe—yet, their impact on user satisfaction are yet to be evaluated.

5.4.6 Perception and Knowledge Conversion

Once the coordinated view has been set up, novices can study and interact with the visualized data with the goal of increasing their knowledge or solving specific tasks. This phase is referred to as internalization Wang et al. (2009) (cf. Fig. 5.3-9). As stated in van Wijk (2005), the amount of knowledge gained depends on the kinds of visual representations used, the users' prior knowledge and their perceptional capabilities. Thus, we took care to address three requirements in our workflow to enhance the internalization process.

To foster **Perception and Internalization** (cf. Fig. 5.3-9), we follow two approaches. First, as for every interactive application, the user interface and interaction design of the visualization platform must be geared towards novices. As an example, this includes self-descriptive and intuitive mechanisms to configure the coordination as discussed above (cf. Sect. 5.4). Based on our prototypes and a number of small user studies, we are continuously working on these issues towards an elaborated user study. Second, beside the platform itself, the visual representations, i.e., the resulting composite application plays the key role for successful internalization. Thus, we consider the users' contexts in the recommendation algorithm of visualization components (cf. Sect. 5.4) to offer him the most suitable and understandable graphical representations. Contextual triggers for this are basic user properties (age, mother tongue, disabilities), preferences, usage (mobile vs. stationary) and device characteristics (e.g., the available screen real estate).

Knowledge Tracking and Externalization (cf. Fig. 5.3-10) is a background task that actively supports several phases of the Information Visualization workflow. As Fig. 5.3 shows, its purpose is to extract implicit visualization knowledge in every

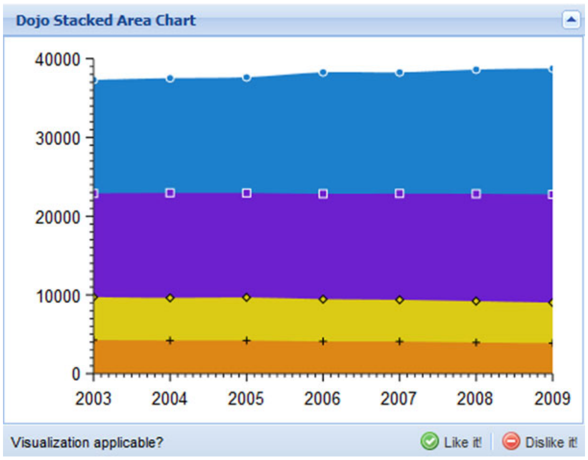

Fig. 5.9 Visualization component with rating bar at the bottom

workflow step to support the upcoming phases. In the following, we briefly present possibilities we see and use in this regard.

Augmentation It is worth considering an expert review step for the data augmentation (cf. Sect. 5.4). By checking and updating automatically generated annotations, their knowledge is externalized, which can be the input for further formalization. Here, the proposed the usage of declarative annotation rules proves its value as they (1) are independent of a special dataset and (2) can be revised by Semantic Web experts without programming.

Reduction As mentioned in Sect. 5.4, we apply association rule mining based on the navigation trails of users in their data sets during the data pre-selection phase. This way, users' understanding of entities and relations of the data is externalized.

Data and Visualization Selection Association rule mining can be employed here, as well. Selected facet values, their assignment as mandatory or optional search criteria, and the previewed visualization components can provide insights into the users' understandings and goals, and they can used to enhance the selection step, e.g., by ordering or highlighting suitable facet values.

Configuration Tracking and externalization in the configuration stage includes the analysis of chosen data mappings to a component, and of user-established coordination patterns between components.

Perception Finally, we use a mechanism to rate single combinations of data and visualization components. Therefore, we distinguish between implicit and explicit ratings. The former are deduced automatically based on the usage of a component in different usage contexts, e.g., a *repeated use* leads to a higher implicit rating while having only a *glimpse* on the component after the integration lowers the rating. The explicit rating is given manually by users clicking "Like" or "Dislike" buttons added beneath every component (cf. Fig. 5.9). Using a collaborative filtering algorithm we then exploit this knowledge to improve recommendations (cf. Sect. 5.4).

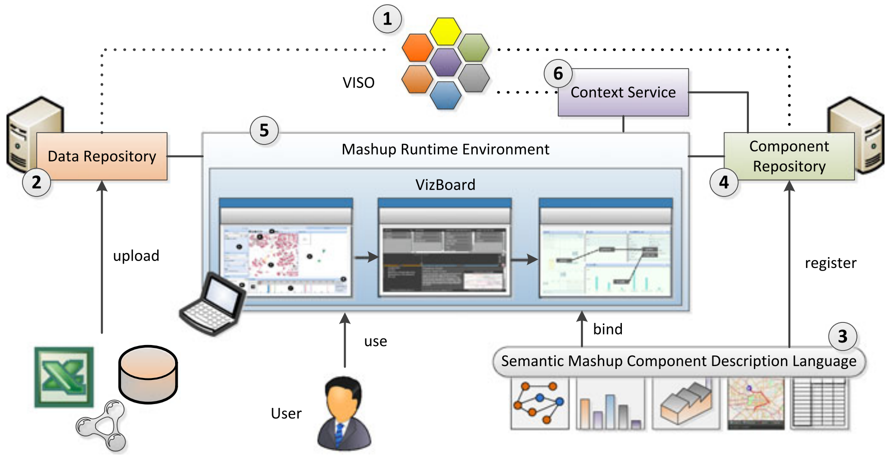

Fig. 5.10 Overview of the architecture of VizBoard

Having presented all the steps of our semantics-driven, context-aware visualization workflow for Semantic Web data, the following section covers our approach of a corresponding software architecture to cover the whole process.

5.5 Component-Based Software Architecture

We specified a component-based software architecture for our visualization system—called VizBoard—according the requirements which come along with our user-centered, semantics-driven InfoVis workflow. As already mentioned in Sect. 5.3, we are building on the CRUISe ecosystem which allows for the dynamic composition of web applications from distributed building blocks. Figure 5.10 gives an overview of the architecture of VizBoard. It comprises six primary parts. In comparison to the architectural overview of CRUISe (cf. Fig. 5.1) we added the visualization-specific vocabulary (1) defined by the **VISO**, which is the glue between the data and visualization components, and the **Data Repository** (DaRe) (2) which provides a common data layer. In the following, we describe the DaRe and the extension made within the CRUISe ecosystem (3)–(6) in more detail. Then, in Sect. 5.5 we give a brief overview of some implementation details.

5.5.1 Data Repository

CRUISe allows for the composition of any data source, i.e., web service, with any user interface widget as long as they are compatible according their semantic de-

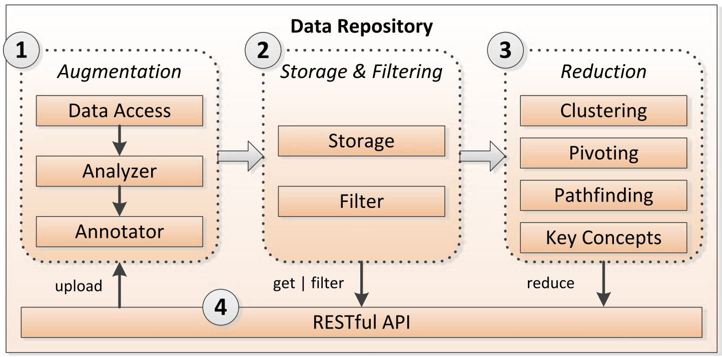

Fig. 5.11 Separation of concerns within the Data Repository

scription of the API. Since its also a desired behavior for the InfoVis domain, it is not applicable in the same way due to the following reasons.

Common Data Access The data to visualize arises from different sources, e.g., web service, files, or databases, and may vary in their formats. Therefore, components are required which allow for a common data access and hide data format specifics like the connection or the query format.

Augmentation and Reduction The aforementioned workflow comprises essential functionality to augment and reduce the data. Our prototypical test demonstrates clearly that most of these functionalities are time consuming and not applicable during the runtime so that a preprocessing is required. Further, the data needs to be available for the asynchronous management of the annotations by an expert.

Performance and Scalability Another problem is the varying performance of the data sources. An own storage layer provides a stable performance and allows to scale on demand.

Security Also if its out of scope, a common data layer enables the management of access rights but also to use security mechanism, e.g., to prevent SQL injections.

For this reasons, a common data layer for all visualizations components, the **Data Repository**, is integrated. Its main components are presented in Fig. 5.11. They could be distinguished into four blocks according their functionality needed within the workflow. Hence, the data access, the analyzers, and the annotator are forming a building block (1) to augment and thus to prepare the data for the following process steps. Subsequently, the homogeneous data is stored within a triple store which allows to request but also to filter the data (2). These functions are required for instance during the integration of the visualization components (cf. Sect. 5.4). To facilitate the data reduction (cf. Sect. 5.4) the components in block (3) query and process the data from (2) on-demand. In the end, the DaRe offers a RESTful web service interface (4) to allow for a platform-independent data access.

5.5.2 CRUISe Extensions

To enable the visualization of Semantic Web data according the proposed workflow we also had to extend the CRUISe architecture. First, **Visualization Components** (cf. Fig. 5.10-3) are specialized user interface components focusing on presentation but mostly neglect other "CRUD" functions like the creation of new data. Thus, we can rely on the component model of CRUISe without any difficulties but the SMCDL is extended to describe visualization features with VISO concepts, e.g., the kind of graphic representation. As the properties as well as the parameters of operations and events are already semantically typed, we can simply point to their generic, semantics-based description of the data structure (Voigt et al. 2012a).

Also the **CoRe** (cf. Fig. 5.10-4) is extended to allow for the semantic-driven management of visualization components based on the enhanced SMCDL. Further, the recommendation for appropriate components (cf. Sect. 5.4) need to be integrated as a multi-level process comprising the discovery and ranking (Voigt et al. 2012a). And some of the knowledge externalization functionalities (cf. Sect. 5.4) are added, especially the collaborative filtering approach for the user-based rating which is employed during the recommendation.

The **MRE** (cf. Fig. 5.10-5) provides the interface between the user, DaRe, and CoRe. Therefore, the complete user-driven workflow is implemented as wizard-like composite CRUISe application comprising specialized and thus, efficient user interface components for each stage. To enable novices to create, edit, and delete communication connections between visualization components, we added a more abstract coordination layer and a helpful meta visualization to show existing communication relations (cf. Fig. 5.8). Further, the runtime is extended to handle RDF data received from the DaRe as shared data layer for all components.

Finally, we utilize the CroCo context service (cf. Fig. 5.10-6) to track the user and device context. We slightly extended its knowledge model using the VISO vocabulary to store visualization specific preferences.

5.5.3 Implementation

All the parts presented in the architectural overview are prototypical implemented to allow for an evaluation. Only single features, e.g., the association rule mining within the data and visualization selection to extract implicit knowledge (cf. Sect. 5.4), are missing. In the following, we highlight some of the implementation details of the DaRe and CRUISe on the whole.

The DaRe is implemented using Java and is accessible through a RESTful web service API using Java Jersey.[12] Its core is a RDF triple store which allows to store

[12]http://jersey.java.net/.

and filter the datasets. To identify an appropriate one, we conducted a benchmark using different real-world datasets on freely available triple stores (Voigt et al. 2012c). Although no store stands out in this test, we decided on Jena TDB[13] due to the existence of an extendable rule engine required for the analysis within the augmentation step. We implemented various data access components to use RDF datasets from uploaded files or received from web services. Further, we integrated Apache POI[14] and the D2RQ engine[15] to use Excel spreadsheets and MySQL databases as data sources. To carried out the data reduction we implemented and integrated numerous algorithms. For example, we employ the JGraphT[16] library for graph calculations, we reuse the key concept extraction API,[17] and integrated RapidMiner 5.2[18] including the RMonto plug-in (Potoniec and Ławrynowicz 2011) to cluster the Semantic Web data.

As aforementioned, we are relying on the CRUISe ecosystem to enable the user-centered, semantics-driven visualization of Semantic Web data. Therefore, we had to extend the existing implementation mainly at three points. First, we integrated the algorithm to recommendation visualization components into the Java-based CoRe. In this regard, we had to extend the semantic model, which stores the information about the registered components, according the visualization specifics from VISO. Our algorithm makes use of the already integrated Apache Jena API and its SPARQL functionality. Second, we developed CRUISe user interface components to implemented the user-centered visualization workflow (cf. Fig. 5.3), e.g., the data pre-selection, and to visualize the data. Like other components, we rely on HTML, JavaScript—using frameworks like D3.js[19] or jQuery[20]—and partly on Adobe Flash. Third, we extended the JavaScript-based MRE to enable the user to create and manage the coordination behavior between components on runtime. To visualize the connections in the meta-visualization we rely on the Raphael library.[21]

The mashup paradigm coming along with CRUISe allows to easily extend our system with new visualization components but also the DaRe is adaptable to use other data sources. All in all, our web-based visualization system VizBoard could be adapted on current needs and thus is applicable on different devices in various domains.

[13]http://jena.apache.org/documentation/tdb/.

[14]http://poi.apache.org/.

[15]http://d2rq.org/.

[16]http://jgrapht.org/.

[17]http://sourceforge.net/projects/kce/.

[18]http://rapid-i.com/.

[19]https://github.com/mbostock/d3.

[20]http://jquery.com/.

[21]http://raphaeljs.com/.

5.6 Conclusion and Future Work

Gaining insights from the growing amount of available Semantic Web data has become seemingly impossible for novices. However, this is exactly the situation that domain experts are facing, as more and more data is provided in the form of RDF, RDFS or OWL. To address this need for user-centered Information Visualization, we have proposed three ingredients: (1) a semantic model formalizing visualization knowledge, (2) a user-centered, semantics-driven visualization workflow utilizing the shared visualization model, and (3) a corresponding software architecture to realize the workflow. While the model has been covered extensively in Voigt et al. (2012a), this chapter has focused on the workflow and its application.

In contrast to existing InfoVis processes, e.g, the pipeline model, our novel visualization workflow actively guides novices from a given set of semantic input data to suitable visualization components using the shared visualization knowledge and contextual information. Even though we have presented a corresponding software system and composition architecture, all steps of the workflow are generic enough to be realized and supported by other tools and frameworks. It should also be noted, that the process itself remains independent from the underlying data models and can thus be employed for arbitrary Semantic Web data. Thus, we facilitate and welcome implementations and evaluations by the community.

As a manifestation of our concepts, we have presented an architecture which implements the workflow and utilizes VISO as the semantic model. To this end, we employ the mashup paradigm whose goal is the combination of existing web resources—in our case RDF data and InfoVis widgets—to create an added value for the user. The architecture is easily extensible with both new visualization components and new data connectors. As it is web-based and includes context knowledge in the composition process, it can be utilized on different devices, such as desktops, tablets, and smartphones, independent of location and time.

As mentioned before, this chapter provides an overview of our work on a user-centered, semantic-driven InfoVis workflow and its implementation in an extensible, open workbench. While the core concepts—the *recommendation* and *selection* of suitable visualization components—has already been validated (Voigt et al. 2012a, 2012b), a high-performance triple store for the DaRe has been identified (Voigt et al. 2012c), and large parts of the workbench have been realized based on the CRUISe platform, a few things remain to be done. In particular, we are planning to conduct three user studies to evaluate our concepts and prototypes of the data reduction (cf. Sect. 5.4.2) and knowledge externalization (cf. Sect. 5.4.6) as well as the acceptance of VizBoard in general.

References

Boukhelifa, N., Roberts, J. C., & Rodgers, P. J. (2003). A coordination model for exploratory multiview visualization. In *Coordinated and multiple views in exploratory visualization* (pp. 76–85).

Brandes, U. (2001). A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, *25*(2), 163–177. doi:10.1080/0022250X.2001.9990249.

Card, S. K., Mackinlay, J. D., & Shneiderman, B. (1999). *Readings in information visualization: using vision to think*. San Francisco: Morgan Kaufmann. ISBN: 1558605339.

Chen, M., Ebert, D., Hagen, H., Laramee, R. S., van Liere, R., Ma, K.-L., et al.(2009). Data, information, and knowledge in visualization. *IEEE Computer Graphics and Applications*, *29*(1), 12–19. doi:10.1109/MCG.2009.6.

Cleveland, W. S., & McGill, R. (1984). Graphical perception: theory, experimentation, and application to the development of graphical methods. *Journal of the American Statistical Association*, *79*(387), 531–554.

Dadzie, A.-S., & Rowe, M. (2011). Approaches to visualising linked data: a survey. *Semantic Web*, *2*(1), 89–124. doi:10.3233/SW-2011-0037.

Ding, L., DiFranzo, D., Graves, A., Michaelis, J., Li, X., McGuinness, D. L., & Hendler, J. A. (2010). TWC data-gov corpus: incrementally generating linked government data from data.gov. In *WWW'10* (pp. 1383–1386). doi:10.1145/1772690.1772937.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, *39*(11), 27–34. doi:10.1145/240455.240464.

Glimm, B., Hogan, A., Krötzsch, M., & Polleres, A. (2012). Owl: yet to arrive on the web of data? In *Linked data on the web (LDOW2012)*.

Grammel, L., Tory, M., & Storey, M.-A. (2010). How information visualization novices construct visualizations. *IEEE Transactions on Visualization and Computer Graphics*, *16*, 943–952.

Haber, R., & McNabb, D. A. (1990). Visualization idioms: a conceptual model for scientific visualization systems. In *Visualization in scientific computing* (pp. 74–93).

Hearst, M. A. (2009). *Search user interfaces*. Cambridge: Cambridge University Press.

Heer, J., van Ham, F., Carpendale, S., Weaver, C., & Isenberg, P. (2008). *Creation and collaboration: engaging new audiences for information visualization* (pp. 92–133). Berlin, Heidelberg: Springer. doi:10.1007/978-3-540-70956-5_5.

Kadlec, B. J., Tufo, H. M., & Dorn, G. A. (2010). Knowledge-assisted visualization and segmentation of geologic features. *IEEE Computer Graphics and Applications*, *30*(1), 30–39. doi:10.1109/MCG.2010.13.

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., & Giannopoulou, E. (2007). Ontology visualization methods—a survey. *ACM Computing Surveys*, *39*(4), 10. doi:10.1145/1287620.1287621.

Leida, M., Afzal, A., & Majeed, B. (2010). Outlines for dynamic visualization of semantic web data. In *LNCS: Vol. 6428. On the move to meaningful internet systems: OTM 2010 workshops* (pp. 170–179). Berlin: Springer.

Mazumdar, S., Petrelli, D., & Ciravegna, F. (2012). Exploring user and system requirements of linked data visualization through a visual dashboard approach. *Semantic Web Journal*. doi:10.3233/SW-2012-0072.

Peroni, S., Motta, E., & d'Aquin, M. (2008). Identifying key concepts in an ontology, through the integration of cognitive principles with statistical and topological measures. In *LNCS: Vol. 5367. The semantic web* (pp. 242–256). Berlin: Springer.

Pietschmann, S. (2009). A model-driven development process and runtime platform for adaptive composite web applications. *International Journal on Advances in Internet Technology*, *4*(1), 277–288.

Pietschmann, S., Tietz, V., Reimann, J., Liebing, C., Pohle, M., & Meißner, K. (2010). A metamodel for context-aware component-based mashup applications. In *Proc. of the 12th int. conf. on information integration and web-based applications & services*.

Popov, I., Schraefel, M., Hall, W., & Shadbolt, N. (2011). Connecting the dots: a multi-pivot approach to data exploration. In *International semantic web conference*.

Potoniec, J., & Ławrynowicz, A. (2011). RMonto: ontological extension to RapidMiner. In *10th international semantic web conference (ISWC2011)*.

Sahoo, S. S., Halb, W., Hellmann, S., Idehen, K., Thibodeau, Jr. T., Auer, S., et al. (2009). *A survey of current approaches for mapping of relational databases to RDF*. W3C RDB2RDF Incubator Group.

Shneiderman, B. (1996). The eyes have it: a task by data type taxonomy for information visualizations. In *Proc. of IEEE symp. on visual languages* (pp. 336–343). doi:10.1109/VL.1996.545307.

Sicilia, M. A., Rodríguez, D., García-Barriocanal, E., & Sánchez-Alonso, S. (2012). Empirical findings on ontology metrics. *Expert Systems with Applications*, *39*(8), 6706–6711. doi:10.1016/j.eswa.2011.11.094.

Tietz, V., Blichmann, G., Pietschmann, S., & Meißner, K. (2011). Task-based recommendation of mashup components. In *Proc. of the 3rd intern. workshop on lightweight integration on the web (ComposableWeb 2011)*. Berlin: Springer.

van Wijk, J. J. (2005). The value of visualization. In *Proceedings of IEEE visualization* (pp. 79–86). doi:10.1.1.75.6547.

Voigt, M., & Polowinski, J. (2011). *Towards a unifying visualization ontology* (Tech. Report No. TUD-FI11-01). Dresden, Germany, TU Dresden. ISSN: 1430-211X.

Voigt, M., Pietschmann, S., Grammel, L., & Meißner, K. (2012a). Context-aware recommendation of visualization components. In *Proc. of the 4th intern. conf. on information, process, and knowledge management (eKNOW 2012)*.

Voigt, M., Werstler, A., Polowinski, J., & Meißner, K. (2012b). Weighted faceted browsing for characteristics-based visualization selection through end users. In *Proc. of the 4th symp. on engineering interactive computing systems*, Copenhagen, Denmark (pp. 151–156). doi:10.1145/2305484.2305509.

Voigt, M., Mitschick, A., & Schulz, J. (2012c). Yet another triple store benchmark? Practical experiences with real-world data. In *Proc. of. the 2nd intern. workshop on semantic digital archives (SDA)*.

Voigt, M., Pietschmann, S., Meißner, K. (2012d). Towards a semantics-based, end-user-centered information visualization process. In *Proc. of the 3rd international workshop on semantic models for adaptive interactive systems (SEMAIS 2012)*.

Wang, X., Jeong, D. H., Dou, W., Lee, S.-W., Ribarsky, W., & Chang, R. (2009). Defining and applying knowledge conversion processes to a visual analytics system. *Computers & Graphics*, *33*(5), 616–623.