Using Expert and Empirical Knowledge for Context-aware Recommendation of Visualization Components

Martin Voigt*, Martin Franke†, and Klaus Meißner*
* *Senior Chair of Multimedia Technology*
† *Software Engineering of Ubiquitous Systems,*
Technische Universität Dresden
Dresden, Germany
Email: martin.voigt,martin.franke,klaus.meissner@tu-dresden.de

***Abstract*—Although many valuable visualizations have been developed to gain insights from large datasets, selecting an appropriate visualization for a specific dataset and goal remains challenging for non-experts. In this article, we propose a novel approach for knowledge-assisted, context-aware visualization recommendation. Therefore, both semantic web data and visualization components are annotated with formalized visualization knowledge from an ontology. We present a recommendation algorithm that leverages those annotations to provide visualization components that support the users' data and task. Since new visualization knowledge is generated while working with a visual analytics system due to users insights, particularly a component is suitable or not for a selected dataset, we track these findings by means of users explicit and implicit ratings. This empirical visualization knowledge is reused in subsequent recommendations to better adapt the ranking of components to users needs. We successfully proved the practicability of our approach by integrating it into a mashup-based research prototypes called VizBoard.**

Keywords-visualization, recommendation, ontology, knowledge, collaborative filtering, mashup

I. Introduction

Through this article we detail, update, and extend our approach for a context-aware recommendation of visualization components [1] presented at the eKNOW 2012[1].

Visualization is a powerful way of gaining insight into large datasets. Therefore, a myriad of visualizations have been developed in recent decades. To bridge the gap between data and an appropriate visual representation, models like the visualization pipeline [2] have been established in numerous tools. As one part of this process, the mapping of data to a graphic representation is critical because only small subsets of existing visualization techniques are expressive and effective for the selected data in a specific context. Generally, domain-specific data can be visualized either using tools which were developed specifically for that domain and use case, or using generic visualization systems. The development of the former requires extensive knowledge by visualization and domain experts, and is usually costly and time-consuming. Thus, in many cases generic visualization tools are preferable, because they are quickly available and reusable in different contexts. Using such tools, domain experts can directly get the information they need out of their data. However, these tools typically require them to select the visualization type and to specify the visual mappings, which can be difficult because they often lack the necessary visualization knowledge [3]. Knowledge-assisted visualization can address this problem by representing and leveraging formalized visualization knowledge to support the user [4]. Suggesting automatically generated visualizations to the user is one promising approach to aid domain experts in constructing visualizations [3], [5].

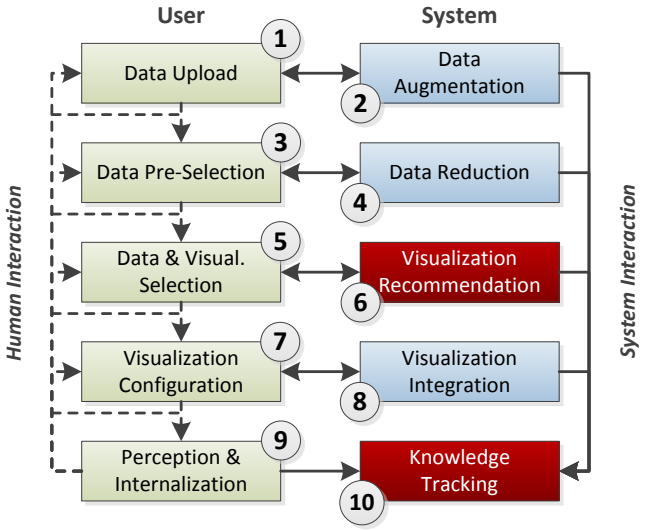

Figure 1. Overview of the complete semantics-based information visualization workflow. The highlighted Visualization Recommendation and Knowledge Tracking are presented in this article in detail.

The concepts defined within this article are essential parts of a semantics-based information visualization workflow for end-users tailored to semantic web dataset [6]. Fig. 1 gives an overview. It consists of five stages users needs to pass: choosing or uploading a dataset ①, getting an overview of the data and choosing a subset ③, selecting relevant data

[1]http://www.iaria.org/conferences2012/eKNOW12.html

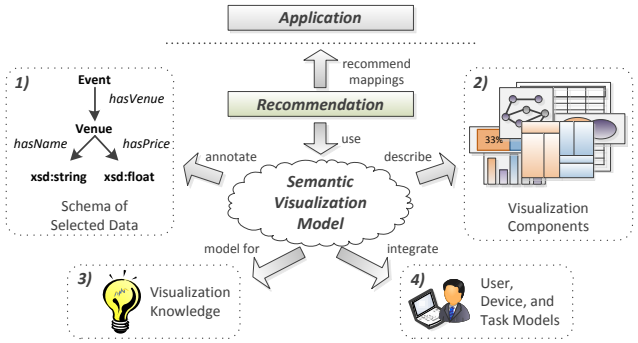

Figure 2. Overview of our goal to recommend mappings of a data source (1) to a visualization component (2) based on a semantic model utilizing visualization knowledge (3) and context information (4).

variables and suitable visualization components ⑤, configuring them ⑦ and, finally, interacting with the rendered data to gain the desired insights ⑨. Due to the interactive nature of the visualization process, users can sequentially pass through, but may 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 (the right rectangles in Fig. 1) where the concepts for the visualization recommendation ⑥ and the knowledge externalization ⑩ are proposed within this article.

We employ a simple example to get our approach across (Fig. 2). It considers a semantic web dataset comprising a list of events hosted at different venues with varying fees. A business user with less visualization experience wants to get an overview of how expensive the events are using his laptop. Thus, he selects a subgraph from a semantic dataset as shown in ① containing two classes (EVENT, VENUE) linked by a Property (*hasVenue*) and two Data Properties (*hasName*, *hasPrice*). To map this data to a compatible visualization component ②, a user needs visualization knowledge ③. Context information ④ about the user (knowledge, skills), his device (hard- /software capabilities) and his task (get overview) must also be considered to create a successful mapping. We strive for a generic recommendation approach utilizing and understanding these different ingredients based on a common semantic knowledge model to facilitate the automated visualization process for different tools.

Our goal of creating a knowledge-assisted, context-aware system which recommends visualization components involves basically five challenges, which are addressed by this article. Firstly, a **formalized vocabulary** for the interdisciplinary visualization domain is required. To this end, we have developed a modular visualization ontology called VISO. Secondly, means to **semantically describe visualization characteristics** of both data sources and visualization components must be provided. Therefore, we propose the linking and annotation of semantic web data and component descriptors with concepts of VISO. Thirdly, appropriate visualization components must be **discovered** for a certain set of data. Thus, we present a matching algorithm which takes the aforementioned formalized visualization knowledge and given user requirements into account to search for compatible visualization components. Fourthly, component candidates need to be **ranked** with regard to the user, usage and device context. Hence, we have developed a corresponding ranking algorithm for the mappings, i. e., component candidates resulting from the discovery. Fifthly, internal visualization knowledge created by the user during the visualization workflow needs to be externalized and **reused**. Accordingly, we defined an architecture and algorithms to externalize, consolidate, store, and reuse this knowledge.

The remainder of this article is structured as follows. First, we discuss related work in the fields of automated or knowledge-assisted visualization, semantic models for visualization, and semantics-based component recommendation in Section II. Then, Section III introduces our visualization ontology VISO in detail and clarifies how it is applied to describe visualization components and data sources. Afterwards, we present the corresponding recommendation algorithm separated into matching and ranking in Section IV. We detailed the gathering and the reuse of empirical visualization knowledge in Section V. Section VI gives an overview of the architecture and its corresponding prototypical implementation. Finally, we conclude the article and outline future work in Section VII.

II. RELATED WORK

The recommendation algorithm presented in this article builds on previous research in the four different research areas: (1) automated visualization, (2) semantic visualization models, (3) mechanisms for semantics-based component discovery and ranking, and (4) collaborative filtering. We will now discuss the state of the art in those four areas.

A. Automated, Knowledge-assisted, and Component-based Visualization

Several automatic visualization systems have been developed to help users to create visualizations. They produce visualization specifications based on user-selected data and implicitly or explicitly represented visualization knowledge. We distinguish between data-driven, task-driven, and interaction-driven approaches. Furthermore, we differentiate into two orthogonal facets: knowledge-assisted and component-based visualization. The firsts objective is to overcome the burden of learning complex visualization techniques by formalizing and sharing domain and visualization

knowledge [7]. We narrow the definition so that these systems need to build on current semantic web technologies, e. g., RDF or OWL, to formalize, use, and maybe share this knowledge in a standardized and widely-adopted way. Component-based visualization means that the single visualization techniques are encapsulated in components or even (user interface) services. It allows for their flexible, context-aware reuse in different scenarios.

Data-driven approaches analyze the meta-model of the data and potentially instance data to generate visualization specifications. Mackinlay addressed the problem of how to automatically generate static 2D visualizations of relational information in his APT system [8]. It searches the design space of all possible visualizations using expressiveness criteria and then ranks them using effectiveness criteria. The more recent visualization mosaics approach from MacNeil and Elmqvist [9] works the same way. Gilson et al. developed an algorithm that maps data represented in a domain ontology to visual representation ontologies [10]. Their visual representation ontologies describe single visualization components, e. g., tree maps. A semantic bridging ontology is used to specify the appropriateness of the different mappings. Our automated visualization approach is similar to the one by Gilson et al. in that both data and visualization components are described using ontologies. The main limitation of data-driven approaches is that they do not take other information such as the user's task, preferences or device into account. Task-driven and interaction-driven approaches usually build on the data analysis ideas present in data-driven approaches, but go beyond them.

The effectiveness of a visualization depends on how well it supports the user's task by making it easy to perceive important information. This is addressed by **task-driven approaches**. Casner's BOZ system analyzes task descriptions to generate corresponding visualizations [11]. However, BOZ requires detailed task descriptions formulated in a structured language and is limited to relational data. The SAGE system by Roth and Mattis extends APT to consider the user's goals [12]. It first selects visual techniques based on their expressiveness, then ranks them according to their effectiveness, refines them by adding additional layout constraints (e.g., sorting), and finally integrates multiple visualization techniques if necessary. In constrast to SAGE and BOZ, our algorithm is ontology-based to allow for reasoning and it leverages device and user preference information.

Visual data analysis is an iterative and interactive process in which many visualizations are created, modified and analyzed [3]. **Interaction-driven approaches** consider either the user interaction history or the current visualization state to generate visualizations that support this process. Mackinlay et al. have developed heuristics that use the current visualization state and the data attribute selection to update the current visualization or to show alternative visualizations

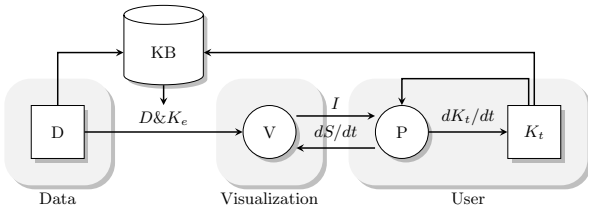

Figure 3. Knowledg-assisted Visualization, based on [15], [16].

[13]. Behavior-driven visualization recommendation monitors users' interactions with visualizations, detects patterns in the interaction sequences, and infers visual tasks based on repeated patterns [14]. The current visualization state and the inferred visual task are then used to recommend more suitable visualizations. Interaction-driven approaches leverage implicit state information, but they consider neither task information that is explicitly expressed by the user, nor user preferences or device constraints.

As mentioned before, **knowledge-assisted visualization** is a more orthogonal aspect of a visualization approach. It means, that the single process steps, e. g., filtering, automated mapping, and configuration, are underpinned by semantic models, i. e., ontologies, which allow for reasoning but also for sharing and, thus, enhancing the knowledge in a collaborative manner. Chen et al. [4] give a good but only a theoretical impression how such a visualization workflow could be designed.

Furthermore, Wang et al. [15] propose a knowledge conversion process in visual analytics system. Fig. 3 sketches this workflow for the identification of the applicable visualization for the chosen data (D). The visualization component (V) represents its (interactive) image I to the user. With its perception (P), the user gets internal knowledge about this visualization over time dK_t/dt (*internalization*) and an insight whether this visualization is applicable or not. If it is not suitable, the user can adapt the representation by changing the specification of the visualization over time (dS/dt). In the end, the internal knowledge (K_t) can be *externalized* to a (global) knowledge base (KB). In case of choosing the same or equivalent data next time, the externalized knowledge is fetched and the suitable visualization presented. Since this theoretical process is valuable for our work, their prototype do not rely on visualization-specific knowledge and, hence, does not cover the recommendation and automated mapping of graphic representations.

Also, the already sketched data-driven approach from Gilson et al. [10] is a knowledge-assisted one since it makes extensive use of ontologies to identify suitable mappings from data to visualization techniques. Its greatest drawback is the manual definition of effective mappings from data items to visual variables within the semantic bridging ontology. In [17], Shu et al. present an ontology (cf. Sect. II-B) and a simple discovery approach for visualization service which regrettably neglects an automated mapping.

Using encapsulated **components** or widgets is a common concept to reuse generic visualization techniques in different contexts or applications. In particular, web-based applications like sgvizler [18] or dashboards [19] employ visualization libraries, e. g., Google Chart Tools[2] or Highcharts JS[3], to present the data. Their biggest disadvantage is that the user has to manually define the mappings. Further, dashboards are mostly static. In contrast, mashup environments like DashMash [20] take the user by the hand to specify mappings or to interconnect the widgets without high cognitive efforts. But all in all, they are not tailored to the needs of automated information visualization since their components are often data-dependent and do not explicitly support the steps of a visualization process. In the visualization domain, some approaches, e. g., [3], [9], are component-based and allow for a flexible combination of visualization widgets. Although they support visualization-specific features like automated mapping or linking and brushing, their components are not loosely coupled like in mashup platforms nor consider context parameters.

In summary, while our work builds on many ideas from automated and knowledge-assisted visualization approaches, in particular the work by Gilson et al. [10] and Wang et al. [15], it is extensible in terms of visualization components, and it considers task, user preferences and device capabilities. In contrast to generative approaches [8], [11]–[13], the strength of using visualization components is that they are optimized for the visual metaphor they represent.

B. Formalizations of Visualization Knowledge

As shown in the previous section, automated visualization requires one or more models to bridge the gap between data and suitable graphic representations. In this regard, prevalent approaches use different concepts, such as rules [12], heuristics [13], and semantic models [10]. We share the view of Gilson et al. [10] that semantic technologies are the methods of choice today. They allow for capturing and formalizing expert knowledge in a readable and understandable manner for humans as well as machines. Therefore, they provide an effective solution for automated recommendation. Further, the current technologies facilitate an easy and dynamic reuse of existing semantic models in new scenarios.

Actually, only few academic works have explored semantic web technologies as means to capture visualization knowledge for describing and recommending resources. Duke et al. [21] were the first proposing the need for a visualization ontology. Their promising approach captures an initial set of concepts and relations of the domain comprising data, visualization techniques, and tasks. Potter and Wright [22] combine formal taxonomies for hard- and software capabilities, sensory experience as well as human actions to characterize a visualization resource. Similarly, Shu et al. [17] use a visualization ontology to annotate and query for visualization web services, with regard to their (1) underlying data model and (2) visualization technique. While the former is a taxonomy comprising various kinds of multidimensional datasets, the latter builds on the data module to classify the graphic representations. For our work, their data taxonomy is not flexible enough as we need to support graph-based data structures for example. Gilson et al. [10] employ three dedicated ontologies to allow for automatic visualization: the first one captures domain semantics and instance data to visualize; the second one describes a particular graphic representation; the final ontology contains expert knowledge to foster the mapping from domain to visualization concepts. In contrast, we allow for a more flexible and generic linking of both sides by annotating each with VISO concepts instead of the explicit, manual creation of an additional ontology. But we reuse this concept of a mapping ontology in a particular way. The adapted concept gives the possibility for an automated insertation of user-generated knowledge by storing mappings of chosen data with the applied visualizations. Rhodes et al. [23] aimed to categorize, store and query information about software visualization systems using a visualization ontology as the underlying model. Their approach facilitates methods for specifying data, graphic representation, or the skill of users.

All in all, we share the goal of the works presented above: defining a formalized vocabulary to describe and recommend visualization resources. However, as we strive for a context-aware recommendation we need a more comprehensive and detailed model that covers not only data and graphical aspects, but also represent the user, his activity, and device.

C. Semantics-Based Component Discovery and Ranking

When it comes to finding and binding adequate services for a desired goal, such as visualizing semantic data as we are, *Semantic Web Services* (SWS) tackle a very similar problem. SWS research provides solutions for finding a service or service composition that fulfills a goal or user task based on certain instance data. Therefore, they employ a formal representation of the services' functional and non-function semantics – usually based on description logics – to facilitate reasoning. Based on this, they strive for the automation of the service life-cycle including the discovery, ranking, composition, and execution of services through proper composition environments.

The discovery of suitable semantic services employs either complete semantic service models, e. g., in OWL-S [24] and WSMO [25], or semantic extensions to existing description formats, as proposed by SAWSDL [26] and WSMO-Lite [27]. The former *top-down* approaches are usually very expressive, but descriptions are complex and time-consuming to build. The latter *bottom-up* approaches add semantic annotations, i. e., references to concepts in external

[2]https://developers.google.com/chart/

[3]http://www.highcharts.com/

ontologies, to WSDL. Even though the above-mentioned solutions cannot be directly applied to our problems, e. g., due to their limitation to web services formats and design principles (stateless), we follow the idea by extending a mashup component description language with semantic references. Thereby, visualization components can be described regarding their data, functional and non-functional semantics, including references to formalized visualization knowledge.

In SWS discovery, suitable services are searched based on a formalized goal or task definition, which is usually a template of an SWS description. Thus, the desired data and functional interface is matched with actual service models. The corresponding algorithms either use measures like text and graph similarities, which restricts the applicability to design-time, or determine the matching degree of services, operations, etc., using logic relationships between annotated concepts as in [28]. In contrast to SWS, we follow a data-driven approach, in which semantically annotated data forms the input for the discovery of suitable candidates. The direct generation of SWS goals from a selected dataset is not feasible. Therefore, we individually match data types, functional interface and hard-/software requirements with and between data and visualization components based on shared conceptualizations. Based on this measure, compatible visualization components can be found.

Ranking of service candidates in SWS bears a number of similarities with ranking visualization components for a certain dataset. It is usually based on non-functional properties, such as QoS and context information (user profile, device capabilities). To this end, a number of sophisticated concepts exist, e. g., for multi-criteria ranking based on semantic descriptions of non-functional service properties [25] and for context sensitive ranking [29]. Since these algorithms are rather generic and work on a semantic, non-functional level, they likewise apply to our concept space.

In summary, the discovery and ranking of candidate services for a predefined goal in SWS research follows a similar principle as our work. Yet, its solutions can not be directly applied to our problems. For one, there is a difference in component models, e. g., with regard to statefulness of visualization components. Furthermore, the discovery of visualization components can not be based on predefined, formalized goal descriptions, as it basically depends on semantic data which is annotated with visualization knowledge. For the annotation of visualization components with semantic concepts though, we can apply the ideas of SAWSDL and WSMO-Lite to the component descriptions. To *link* semantic data with visualization components, a shared conceptualization of visualization knowledge is needed. Therefore, the next section presents VISO.

D. Collaborative Filtering Mechanisms

One approach to share and track user-generated knowledge are Collaborative Filtering Recommender Systems (CFRS) [30]. In contrast to the content-based recommendation, which employs the structure of the items, these systems investigate the similarity of ratings for items given by users. Hence, no content analysis or tagging by experts is required. All knowledge is generated due to ratings from end-users while using the system.

The ratings can be distinguished into implicit and explicit ones [31]. Implicit ratings are acquired by tracking the user interactions within the application, by reaching predefined time slices or a number of iteration steps. Nichols et. al [32] specify an extensive list of possible kinds implicit ratings and their corresponding recognition. The most suitable ones for us are *Repeated Use*, *Glimpse* and *Associate Ratings*. In contrast to the implicit ratings, the explicit ones are concretely expressed by the user by a concrete interaction, e. g., by pressing a button. The only requirement is that the user knows what is the effect of the interaction or rating. Unfortunately, it is challenging to identify suitable methods and scales since it is usually a trade-off between getting a detailed opinion from the user by not overburden or scaring him. For example, ebay gathers feedback on four distinct 5-star scales. On the other hand, facebook just eases the interaction by just *liking* content.

The algorithm used for the CFRS is the *Neighborhood-based Recommendation Method*, which can be calculated in two different ways [33]–[35]. The user-based approach looks for similar users in the system, by finding akin ratings according to the actual user. This approach lacks mainly in the possibility to justify the calculated prediction to the user [35]. The system has only the ability to present similar users not items, which is not appropriate for a mainly item-oriented system like ours. The item-based approach searches for similar items to make a prediction of interesting ones for the specific user.

Unfortunately, the recommender systems compete with the cold-start problems, which are based on the non-existence of ratings. According [36], they can be classified into three categories: (1) *new users* or (2) *new items* have no ratings in the system, hence, it is impossible to find similar items or users; (3) a *new community*, which comprises new users and items, is applied. For these problems, the CFRS community has identified different strategies, e. g., un-personalized results, automatically generated ratings, or closed beta phases, which should be considered and carefully balanced during the system design.

To the best of our knowledge, CFRS are not used within automated visualization systems so far. In our opinion, the main reason is that they may identify suitable graphical representations based on users ratings but do not allow for an automated mapping of data structures and values to visual

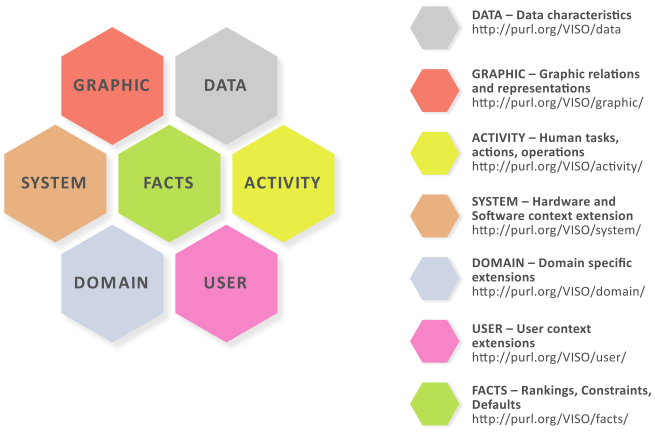

Figure 4. Overview of the VISO.

structures and attributes. However, if this mapping is already provided by the visualization system, collaborative filtering introduces a new facet for recommending visualization techniques: users conclusion if a mapping is suitable in the current context. Thus, it allows for tracking and storing user knowledge – the so-called externalization (cf. Sect. II-A) – and enables the adaption of the visualization process due to this evolving knowledge.

III. VISO: A Modular Visualization Ontology

The foundation of our visualization recommendation approach is a formalized, modular visualization ontology called VISO [37], [38]. It provides a RDF-S/OWL vocabulary for annotating data sources and visualization components, contains factual knowledge of the visualization domain, and serves as a semantic framework for storing contextual information. Altogether, it serves as a *bridging ontology* between semantic data and visualization components by offering shared conceptualizations for all four mapping ingredients shown in Fig. 2. Details of VISO and its development are described in [37], [38]. Furthermore, it can be downloaded and browsed under http://purl.org/viso/. The seven VISO modules (data, graphic, activity, user, system, domain, and facts) represent different facets of data visualization domain. They refer to each other and to other existing ontologies as needed. In the following, we discuss essential parts of the ontology, which are used for the recommendation of visualization components, in detail.

Data: Fig. 5 shows parts of the data module which contains concepts for describing data variables and structures for visualization purposes. While all concepts are employed to describe visualization components, those with dotted lines are also used to annotate semantic data. The vocabulary is especially needed at component-side to describe possible input data in a generic manner as the most visualizations allow for representing domain independent data. For example, a simple table may visualize data about hotels,

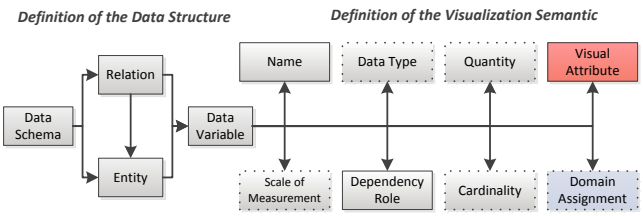

Figure 5. Overview of the VISO data module.

cars, or humans. Using this vocabulary, we specify only the data structure and characteristics. As can be seen, a DATA SCHEMA consists of ENTITY and RELATION concepts. The latter represent links between ENTITY concepts like an OWL Object Property. Both ENTITIES and RELATIONS can contain DATA VARIABLE concepts, whose equivalent in OWL space is a Data Property. For example, the semantic data model of a table visualization component would be represented as one ENTITY concept with several DATA VARIABLES for every column. Further semantics, e. g., the SCALE OF MEASUREMENT and CARDINALITIES – specified using built-in OWL constraints – can be defined on the DATA VARIABLE concepts (Fig. 5) to constrain its scale etc. By linking the concepts from the data module to the VISUAL ATTRIBUTE concepts from the graphic module, we bridge the gap between data attributes and visual elements.

Graphics: The graphics module conceptualizes the semantics of GRAPHICAL REPRESENTATIONS and their parts, e. g., their VISUAL ATTRIBUTES. Concrete graphical representations, e. g., *scatter plot* and *treemaps*, and concrete visual attributes such as *hue* or *shape* are contained as instance data. The concepts from the graphics module are used to semantically annotate visualization components and to define visualization knowledge in the facts module.

Activity: The activity module models user activity in a visualization context. It builds on the ontology-based task model by Tietz et al. [39], which distinguishes betweens high-level, domain specific TASKS and low-level, generic ACTIONS, similar to the distinction made by Gotz and Zhou [40]. We have extended the action taxonomy of Tietz's task model by separating data- and UI-driven ACTIONS, and by formalizing ACTIONS from the visualization literature such as *zoom* and *filter*. This enables the fine-grained annotation of interaction functionality in visualization components.

User: The user module formalizes user PREFERENCES and KNOWLEDGE. Users can, for example, have PREFERENCES for different GRAPHICAL REPRESENTATIONS, and their *visual literacy* can differ. As manifold context models for users, their characteristics and preferences, already exist those can be seamlessly integrated and used here.

System: The system module facilitates the description of the device context, e. g., installed PLUG-INS or SCREEN SIZE. It also allows us to annotate a visualization component

with its system requirements. Again, sophisticated models for device characteristics and context exist, which were reused or integrated in this module. As an example, we borrow concepts from the *CroCo* ontology [41], which combines user, usage, system, and situational context from different existing works developed by academia.

***Domain**:* Many visualizations are domain-specific, and thus it is important to consider the domain context during visualization recommendation. However, it is not feasible to model all possible visualization domains. Instead, we support linking to existing domain ontologies. A DOMAIN ASSIGNMENT links VISO concepts, e. g., a DATA VARIABLE (Fig. 5), to concepts from specific domain ontologies. As this assignment is usually created automatically during data analysis, it can be qualified with a probability value reflecting its accuracy. Thus, the analysis of a data source with ambiguous Properties, such as *typeOfJaguar* and *typeofApple*, will result in multiple domain assignments with probabilities below 1. In contrast, a Data Property *hasPrice* from our motivating example could be annotated with *price* and a probability of 1. A visualization component supporting DATA VARIABLE annotated with the more general concept *value* could be inferred as a possible mapping.

***Facts**:* The visualization recommendation also depends on factual visualization knowledge to select suitable visualizations. Thus, we formalized knowledge from the information visualization community, e. g., verified statements such as "position is more accurate to visualize quantitative data than color" [42], to make it machine-processable. These rankings and constraints are formalized in rules in the Facts module. These rules use of the vocabulary of the other VISO modules in their conditions part, e. g., SCALE OF MEASUREMENT (*quantitative*) and the VISUAL ATTRIBUTE (*position, color*) for the mentioned example. If the conditions are matched, a rating is assigned to the corresponding visualization component description.

To give a more practical insight, the following example explains how a *treemap* visualization is described using VISO (Fig. 6). First, the hierarchical data structure of the *treemap* is specified. At the top level, a *Node* ENTITY represents the whole *treemap*. It can contain *Leaf* ENTITIES and *Node* ENTITIES. The label and size variables of *Leafs* can be configured. They are annotated with visualization semantics, e. g., the SCALE OF MEASUREMENT for the label variable is *nominal* and the ROLE of the size variable is *dependent*. Further domain semantics could be added to the variables, e. g., WordNet (http://wordnet.princeton.edu/) concepts such as *value*. In addition to the data structure and the variables, more general semantics such as the kind of GRAPHIC REPRESENTATION (*treemap*), the LEVEL OF DETAIL (*overview*) and possible ACTIONS (*select*, *brush*) are defined for the entire visualization component.

In order to facilitate the construction of visual mappings,

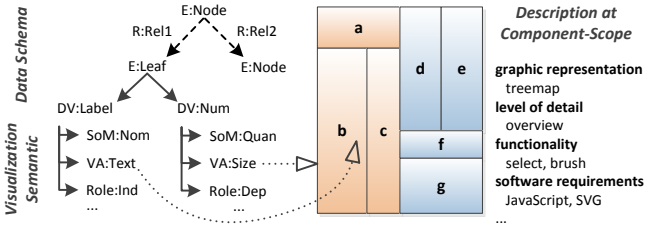

Figure 6. Description of a treemap visualization in VISO.

VISO is used to annotate visualization components and semantic web data. In the latter case, we annotate only RDF Properties on a schema level. RDF Properties hold the data that will be visualized, e. g., literals and relations, whereas RDF classes assemble such properties and do not provide additional information that would be relevant for visualization. Similarly, annotations are made on the schema level, because instance data annotation would be redundant. Consider our motivating example (Fig. 2-1), comprising the Property *hasPrice*. Because the Property has the RDFS Range *xsd:float*, the required DATA TYPE is already defined and the SCALE OF MEASUREMENT is *quantitative*. The number of distinct values (CARDINALITY) and the overall number of values (QUANTITY) can be extracted from the instance data. While a DOMAIN ASSIGNMENT is not mandatory, it could be applied, e. g., to *price* from the WordNet vocabulary.

In summary, VISO models the concepts required for data visualization. It is used to annotate data, to describe visualization components, to represent context and factual knowledge. Together, these different pieces are the foundation of our visualization recommendation algorithm.

IV. VISUALIZATION RECOMMENDATION ALGORITHM

The visualization recommendation algorithm creates an ordered list of mappings of visualizations components for the selected data (Fig. 2-1). It considers contextual information (e.g., device, user model) as well as knowledge about the full data source. While the user model and device are mandatory inputs, visualization specific information like the required LEVEL OF DETAIL or the requested kind of GRAPHICAL REPRESENTATION are optional constraints.

The algorithm consists of two separate steps: matching and ranking (Fig. 7). Both steps leverage semantic knowledge formulated as VISO concepts (cf. Sect. III). In the matching step, potential mappings between data and widgets are generated based on functional requirements. The resulting visualization set is then sorted in the ranking step using the formalized visualization knowledge, domain concepts, and contextual information.

A. Discovery of Mappings

The matching algorithm generates a set of mappings from the selected data to visualization components (Fig. 7).

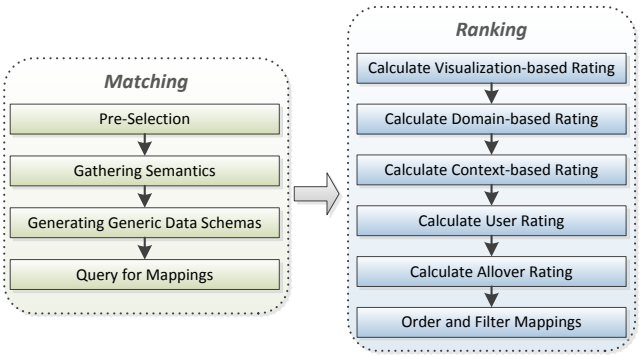

Figure 7. Overview of the recommendation algorithm.

First, potentially applicable widgets are identified and non-applicable components are ruled out (**pre-selection**), since limiting the set of available visualization components early improves the overall algorithm performance. To be applicable, a widget has to (1) be compatible with the target device (e. g., required PLUGINS must be available), (2) support the number of selected Data Properties, and (3) support visualization and task specific requirements (e. g., showing an *overview*), if specified by the user. As can be seen, these constraints do not relate to data structure or semantics of the data variables, yet. Semantic matching is carried out with the resulting component candidates in the following step.

Second, semantics, e. g., the SCALE OF MEASUREMENT, DATA TYPE, and QUANTITY (Fig. 5) of the selected Properties are fetched (**gathering semantics**). For example, the DATA TYPE *xsd:float* or the SCALE OF MEASUREMENT *quantitative* of the property *hasPrice* (Fig. 8-3)) would get retrieved. This semantic information about the Properties is used in the next steps.

Third, we **generate generic data schemas**, which are then used to query for mappings. We distinguish between tabular and graph-based DATA SCHEMAS. TABULAR DATA SCHEMAS contain one ENTITY with several DATA VARIABLES (Fig. 8-1). GRAPH-BASED DATA SCHEMAS contain two or more linked ENTITIES, each containing zero or more variables (Fig. 8-2).

If a **single class** has been selected, a TABULAR DATA SCHEMA is chosen and an ENTITY is created for that class. For every selected Data Property of this class, a DATA VARIABLE with the semantic information (that was retrieved in the previous step) is attached to the ENTITY.

If **several classes** have been selected, we generate both a tabular and a graph-based DATA SCHEMA. For the TABULAR DATA SCHEMA, a single ENTITY gets created. For any selected Data Property from those classes, a DATA VARIABLE with the semantic information is attached to the single ENTITY. This reduces the graph-based data structure to a tabular structure. For example, consider the data shown in Fig. 8-3. The algorithm would create one ENTITY with two DATA VARIABLES. The first DATA VARIABLE would represent the semantics of *hasName*, e. g., the *nominal* SCALE OF MEASUREMENT, and the second DATA VARIABLE would represent *hasPrice*. A GRAPH-BASED DATA SCHEMA gets generated as follows. Beginning with a class from the input data, e. g., *Event* in Fig. 8-3, an ENTITY is created. Similar to the other cases, DATA VARIABLES and their semantics are attached to this ENTITY for the selected Data Properties linked to the class. Next, for each Object Property connected with the class, a RELATION gets generated. If the target class for that RELATION has not been processed yet, it is created and processed in a similar way. This depth-first processing continues until the current part of the input graph is completely traversed. If there are multiple unconnected classes in the input, the algorithm continues with those until all graph components are processed, e. g., the algorithm would generate the DATA SCHEMA illustrated in Fig. 8-4 by processing the input data structure shown in Fig. 8-3.

Fourth, the mappings are generated by querying the semantic representations of the pre-selected components with the generic DATA SCHEMAS that were computed in the previous step (**query for mappings**). The mappings include permutations of DATA VARIABLES with similar semantics, and thus the number of mappings may be higher than the number of existing components. Using the data structure generated by the algorithm for the example shown in Fig. 8-4, both the *scatter plot* (Fig. 8-1) and the *treemap* (Fig. 8-2) would fit on the level of data structure. However, only the *treemap* is a suitable mapping due to the annotated semantics which are also employed by querying. The *scatter plot* is not suitable because it has two *quantitative* DATA VARIABLES, where both a *nominal* and a *quantitative* DATA VARIABLE are required. The generated set of mappings from the selected data to the visualization components is ranked in the next part of the algorithm.

B. Ranking of Mappings

The ranking step of the algorithm sorts the visual mappings that were generated by the previous matching step. While this step identifies valid mappings and visualization components that satisfy functional criteria, it does not take their effectiveness into account. To sort the mappings by their effectiveness, the ranking step applies factual visualization knowledge, domain assignments, contextual user and device information, and a user rating.

The four different kinds of rating are combined using an arithmetic mean. The overall rating has a range between 0 and 1. We weight all three, receptively four, rating types equivalently for two reasons. First, the assignment of a (quantitative) rating is often subjective. Second, a profound user study is needed to evaluate the impact of each knowledge base in users visualization selection process what will be future work. As x, y, z, and r_u are the number of each kinds of rating, the overall rating R for each mapping is

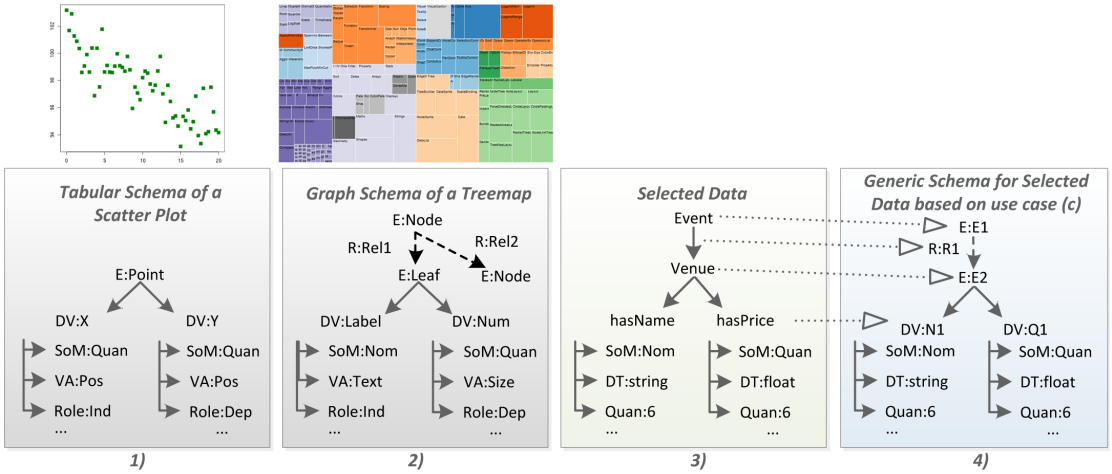

Figure 8. Comparison of the data structure and the annotation between 1) a scatter plot, 2) a treemap, 3) user's selected data, and 4) a generic equivalent of the selected data.

calculated in terms of eq. 1. The meaning factor $1/n$ is therefore assigned with $1/3$, if neither an user rating can be found, nor can be calculated. In all other cases, it is assigned with $1/4$ to keep the equivalently rating of all factors.

1) Factual Visualization Knowledge: The factual visualization knowledge (see Section III) is defined by a set of rules that consists of a condition and a rating. The conditions are specified using the VISO vocabulary for the visualization components. For each widget, the ratings of all rules that are met are added to its specification. During runtime, the arithmetic mean of all ratings r_v is calculated for the discovered component of each visual mapping. For example, we formalized rules to rate the appropriateness of visual encodings for quantitative data [42]. The *quantitative* DATA VARIABLE of the *treemap* (Fig. 8-2) is rated with 0.5 as it employs "only" *size* and not *position*.

2) Domain Assignments: Domain concepts from various ontologies are assigned to both the data input and the visualization components with a certainty value (see Section III). For each pair of input Property and DATA VARIABLE of the visualization component, we calculate a semantic similarity rating between 0 and 1 (e. g., using [43]), if they both have a domain concept assigned with a certainty greater than 0. The final rating r_d is the product of the semantic similarity and the arithmetic mean of both certainties. In our example (see Sect. III), we used *value* and *price* from WordNet to annotate the *quantitative* DATA VARIABLE of the *treemap* and the Property *hasPrice* from our dataset, each with a certainty of *1*. Using [43], we get a rating r_d=0.9094.

3) User and Device Information: The rules for the context-based rating r_c are part of the knowledge base and use the VISO vocabulary, similar to the factual visualization knowledge. The rules are executed during runtime and employ the above mentioned identifiers of users' and their device. For example, we construct a SPARQL-based rule that counts the use of different GRAPHIC REPRESENTATIONS, like *treemaps* or *scatter plots*. This rule assigns a rating r_c between 0 and 1 to the visual mappings.

4) User-shared knowledge: The factor r_u is associated to our concept of employing the user-generated visualization knowledge [15] by using collaborative filtering, see Section V. Since the user has not rated the visualization component in the specific combination with a selected dataset, the algorithm tries to foresee a possible rating. This calculated rating r_u assigns a factor between 0 and 1.

Finally, the complete list of mapping is ordered based on the combined ratings R for each mappings. This ranking could be used to automatically display the most suitable visualization component to the user, or, as in our approach, to let the user pick one of the top n ranked visualizations.

V. REUSE OF EMPIRICAL VISUALIZATION KNOWLEDGE

As identified in Sect. II-A, the idea of *knowledge-assisted visualization* is mostly presented in a theoretical way and lacks of concrete descriptions of data structure or applied algorithms. Hence, in the latter sections we propose the VISO as well as a concrete recommendation and mapping algorithm, which employs the ontological knowledge base, to enhance the so-called internalization process [15]. Unfortunately, we only use the a priori knowledge formalized by experts so far. Users insights, particularly a component is suitable or not for a selected dataset, are neglected. Thus, in the following subsections we provide concepts to externalize and reuse also this empirical knowledge which is lost otherwise.

A. Externalization of Empirical Knowledge

The *externalization* process describes the storage of user's internal knowledge within the system. This process can be distinguished in acquiring *implicit* and *explicit* insights. The

$$R = \frac{1}{n := \{3,4\}} \left(\frac{1}{x} \sum_{i=1}^{x} r_{v_i} + \frac{1}{y} \sum_{j=1}^{y} r_{d_j} + \frac{1}{z} \sum_{k=1}^{z} r_{c_k} \ (+ \ r_u) \right) \quad (1)$$

tracking and interpretation of interactions as knowledge is called *implicit rating* and is used for gathering knowledge, before the user indicates the end of the adaption loop by an *explicit rating*. In our concept, a rating is always saved for a combination of a generic data schema (cf. Sect. IV-A) and concrete visualization component.

For the storage of ratings, we developed a rating ontology similar to the Semantic Bridging Ontology of Gilson et al. [10]. It maps all ratings of one generic data schema with one visualization component, see Fig. 9. The ratings are saved in a flat table beside these combinations. The flattened table is necessary to apply the collaborative filtering algorithm on ontologies without the conversion of the data types. However, the use of an ontology allows for a simple reuse of other concepts within the VISO without duplicating information. Thus, we are able to query for instance "only good rated components that are able to visualize trends". To harvest implicit rating, we rely on the following three actions from [32].

- **Repeated Use:** A visualization component is *used* more than three times by the same user. The usage of the component is recognized by counting interactions within a defined time interval. Since the repeated use is a sign that the user favors a component, it is added to a so-called white list.
- **Glimpse:** If the chosen visualization component is discarded without reaching a defined time interval or a count of interactions for recognizing the *Repeated Use*, it is downgraded by adding it to a black list.
- **Related Rate:** The visualization component was explicit rated by the user, but in a different data combination. Since the user knows its characteristics, it is possible that he likes it for other data selections, which are distinct from the generic data schema, as well. In case of a good rating, the component is added to the white list, otherwise to the black list.

Beside this implicit knowledge tracking, we gather *explicit* ratings. Thus, the user can explicitly decide whether the visualization is applicable for its purpose or not. Since we like to stimulate the user to rating, we employ a simple scale of *applicable* (1) or *not applicable* (0). Thereby, we fulfill the requirement to give the user an adequate possibility to rate, without an excessive demand.

B. Collaboration

The *collaboration* process describes a direct collaboration of two or more users [15], such as chat, telephone, or co-browsing. We broaden this scope by including indirect

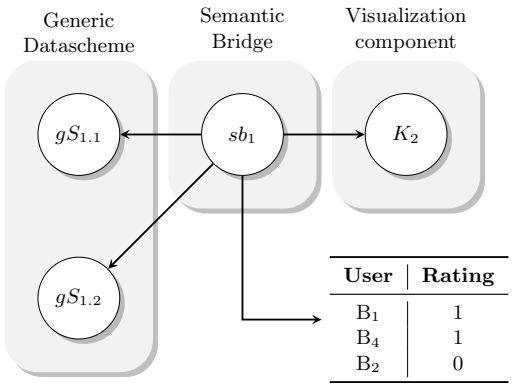

Figure 9. Overview of the rating ontology.

	$\mathbf{K}_1$	$\mathbf{K}_2$	$\mathbf{K}_3$	$\mathbf{K}_4$	$\mathbf{K}_5$
B_1	1	0	-	0	0
B_2	0	1	0	1	1
B_3	0	r_{rs}	1	1	1
B_4	1	1	1	1	-

Figure 10. Example set for CFRS prediction.

sharing of knowledge as collaboration. For this purpose, we use algorithms for the collaborative filtering, what allows for predicting a rating based on similar users and ratings. After an evaluation on accuracy, efficiency, stability, justification, and serendipity [35], we decide to employ the item-based approach for our use-case. It calculates the similarity between the ratings of visualization components, given by different users. With this information, the algorithm can predict the rating for the current visualization. An example is given in Fig. 10. It has to calculate the prediction r_{rs} for user B_3 and visualization component K_2. In this exemplary setting, the prediction is assessed based on the rating distances between K_2 and all other visualization components (K_1 - K_5). The algorithm chooses K_4 and K_5 as *nearest neighbours* and forecasts a rating of $r_{rs} = 1$ for this setting. The formal calculation can be considered in detail in the work from Sarwar et. al [34] and could be used without adaption, concerning the flattened structure of the rating storage.

The quality of the CFRS can be measured by calculating the *mean absolute error*. This technique involves the distance between the predicted and the encountered rating. All distances are saved within the system and are normalized by their count. This comparative simple method can be applied

in recommender systems, that scales the rating on a base of 0 and 1 [44]. If the measured error is near 1, the CFRS will predict the wrong values and, therefore, sharing *wrong knowledge*. In this case, we deactivate the prediction in the ranking algorithm, acquire more ratings and activate it if a given threshold is reached.

If no rating exists and also no one could be predicted due to missing information, our approach tries to employ the implicit ratings stored in the white and black list as mentioned in the foregoing subsection. The problem of their application is the missing expressiveness as it is always vague if the implicit rating is true or not. We decide to assign the weight of 0.25 if a component is on the black list and 0.75 if it is on the white list. Their appropriateness needs to be analyzed and maybe aligned in a broader evaluation scenario.

C. Using Empirical Knowledge for Evaluation

The empirical knowledge, especially the explicit rating is an interesting foundation for the evaluation of our ranking algorithm presented in Sect. IV-B. We can calculate the distance between the mean of *factual visualization knowledge*, *domain assignment* and *user and device information* to the given explicit rating. As it is also normalized to a scale between 0 and 1, it is possible to apply the method of the *mean absolute error*. In case the measured error converges to 0, we can verify that the calculated elements of the ranking algorithm are significantly correct. In contrast, if the measured error converges to 1, the ranking algorithm and the users needs shows a big gap, which can disprove its correctness.

VI. Architecture and its Implementation

After presenting the conceptual foundations of our approach, in this section, we show how they are integrated into a knowledge-based and mashup-based architecture. Furthermore, we outline some implementation-specific details.

A. Knowledge-based Architecture

To realize the concepts discussed above, we specified a reference architecture shown in Fig. 11. It comprises three layers: ontological knowledge bases, loosely-coupled web services, and a component-based user interfaces. In the following, we describe the functionality of the single parts and their relation amongst each other in detail.

The first knowledge base, the *VISO* ① (cf. Sect. III), holds the visualization specific knowledge. It is used to annotate the data within the Data Repository ⑤ and to describe the visualization capabilities and the data interfaces of components. The foundation of the semantic component description is the *Mashup Component Description Ontology* (MCDO), which is part of the CRUISe mashup environment ② [45]. It is not only extended by VISO but also by contextual meta-information. For this, we reuse the *CroCoOn* ontology ③ being part of the CroCo context service ⑦ [46]. Finally, we designed an ontological knowledge base to store *ratings* ④ for the mapping of selected data to the chosen component. Therefore, it refers to concepts of the VISO and MCDO.

Furthermore, we build on four different web services which heavily make use of the mentioned knowledge bases. The *Data Repository* ⑤ offers a homogeneous data layer for the visualization system to upload, convert, filter, and cluster the data. Furthermore, it semi-automatically augments it with VISO vocabulary like described in Sect. III. The *Component Repository* ⑥ – being part of CRUISe as well – allows for the semantic-driven management of visualization components based on the MCDO ②. Further, the recommendation for appropriate components (cf. Sect. IV) is integrated within this service. For this, it gathers data semantics from the Data Repository, the Context Service, and the Rating Repository. As mentioned, we reuse CroCo [46] as *Context Service* ⑦. The *Rating Repository* ⑧ offers the functionality to store all the implicit and explicit ratings of the users tracked in the user interface. Additionally, it provides an API to retrieve this score. If it is not available directly, the algorithm defined in Sect. V is applied to predict this rating.

The visualization workflow of an user is accomplished by VizBoard – a composite application based on CRUISe running within the *Mashup Runtime Environment*. It complies with the process presented in Fig. 1, where the most crucial steps are shown in Fig. 11. After uploading the data, the user has to slice and dice it to a manageable subset by using the *Data Pre-Selection* component ⑨ [47]. This reduced dataset is the foundation for the concrete selection of data items and structures to visualize. Since also visualization-specific characteristics, e. g., visual variables or interaction techniques, are important to select appropriate visualization components, we developed the sophisticated concept of *Weighted Faceted Browsing* [48]. The related component ⑩ access the Component Repository to execute the recommendation algorithm proposed in Sect. IV. In the end, the selected components are integrated ⑪. Thus, the user can perceive the represented data. At this stage, we explicitly and implicitly acquire users knowledge, like explained in Sect. V, and save it using the Rating Repository.

B. Implementation Details

After giving an overview of our architecture, we provide some details on its implementation. All ontologies are build on the widely adopted semantic web standards from the W3C: RDF[4], RDFS[5], and OWL[6]. To define the schemata, we mostly rely an OWL DL, which is expressive, deterministic, and allows for inferring new knowledge by using different

[4]http://www.w3.org/TR/rdf-primer/
[5]http://www.w3.org/TR/rdf-schema/
[6]http://www.w3.org/TR/owl2-overview/

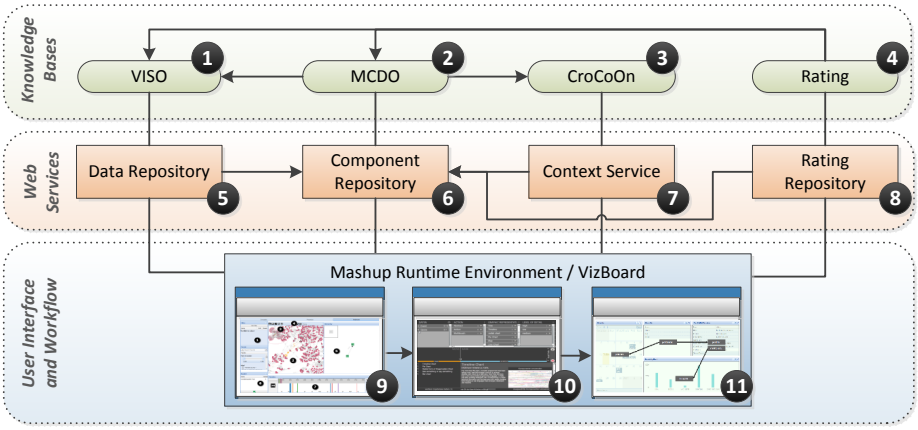

Figure 11. The architecture of our approach could be distinguished into three layers: knowledge bases, web services, and user interface.

reasoners. To query the ontologies within our services, we build on SPARQL 1.1[7].

All web services presented in the architectural overview (Fig. 11) are prototypical implemented in Java. The Data Repository is accessible through a RESTful web service API using Java Jersey[8]. Its core is a RDF triple store which allows to store and filter the datasets. To identify an appropriate one, we had to conduct a triple store benchmark [49] since the existing ones do not consider real-world datasets, SPARQL 1.1, nor reasoning which are altogether requirements in our use case. Although no store stands out in this test, we decided on Jena TDB[9] due to the existence of an extendable rule engine required for the analysis within the augmentation step.

As aforementioned, we are relying on the CRUISe ecosystem. Thus, we could reuse the Component Repository and the Context Service. Since the latter does not require any extension, the recommendation algorithm (cf. Sect. IV) is implemented in the Component Repository using the Apache Jena API and Jena rules. Additionally, we employ SPARQL to pre-select components and to generate the generic data schema like proposed in Sect. IV-A. These queries run against the semantic information of the components stored within the MCDO, particularly the operations of the component API as they are responsible to insert the data.

The Rating Repository, which manages the implicit and explicit ratings for combinations of generic data schemata and components, is accessible over a REST interface, too. To retrieve components and their ratings for a generic data schema, which could be represented as unique hashes, we employ SPARQL as well. List. 1 shows an exemplary query. Furthermore, we implemented the CFRS algorithms defined in Sect. V-B. This allows for not only to use the rating made by the user but also to predict ratings if not available.

```
# get ratings by given generic data schema
SELECT ?vcid ?owner ?val
WHERE {
 ?gds v-r:hasGenericDataScheme
   v-d:1296abf85e507a9596ab2131a0f933a3 .

 ?sbo v-r:hasGenericDataScheme ?gds;
   v-r:hasRating ?ratings;
   v-r:hasVisualizationComponent ?viscomp.

 ?ratings v-r:hasRatingValue ?val;
   v-r:hasOwner ?owner.

 ?viscomp mcdl:hasId ?vcid.
}
```

Listing 1. SPARQL query to retrieve all ratings for a data schema.

At the user interface layer, we use the Mashup Runtime Environment, which is implemented as purely JavaScript-based thin-server architecture and as client server architecture by using Java for the backend and JavaScript for the frontend. Both client-side implementations are extended to allow for voting the data-component-combinations. The implicit rating starts with the loading of a visualization component into the screen (Fig. 11-11). We included all three recognition modes distinguished in Sect. V-A by tracking mouse and key events in a defined time span after components' integration. Furthermore, the runtime automatically integrates rating buttons for every component beneath the graphic representation (Fig. 12).

All user interface components being part of the user-centered visualization workflow, e. g., the Data Pre-Selection, and the components to visualize the data are developed using HTML, JavaScript, e. g., frameworks like D3.js[10] or jQuery[11], and partly Adobe Flash.

[7]http://www.w3.org/TR/rdf-sparql-query/
[8]http://jersey.java.net/
[9]http://jena.apache.org/documentation/tdb/

[10]https://github.com/mbostock/d3
[11]http://jquery.com/

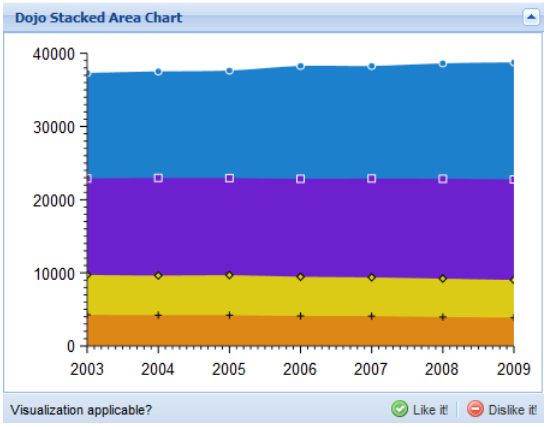

Figure 12. Visualization component with rating bar at the bottom.

VII. Conclusion and Further Work

Selecting an appropriate visualization for a specific dataset in a specific scenario remains challenging for non-experts. Therefore, we have presented a context-aware and knowledge-assisted approach to recommend suitable visualizations for semantic web data. Its foundation is the modular visualization ontology VISO, which provides the vocabulary to annotate both data sources and visualization components. Based on these shared concepts from the visualization domain, our recommendation algorithm covers both matching and context-aware ranking of suitable graphic representations. First, possible mappings from data to visual encodings are identified using the selected data, its semantics, and other functional information. Then, quantitative ratings for each mapping are calculated with respect to visualization knowledge, domain concept relations, context information, and user ratings. To the best of our knowledge, this approach is the first that employs formalized, inferred expert knowledge but also empirical, evolving knowledge from users to identify the most suitable visualization components.

Currently, we are also planning to conduct an exhaustive user study to identify and model the interdependencies between the knowledge bases employed within the ranking. Furthermore, we are working on a concept to use the a priori and empirical knowledge to assist the user in interpreting and understanding the visualized data what will underpin the usefulness of knowledge-assisted visualization.

Furthermore, many concepts presented in this work can be adapted to general semantic reasoning problems. As a goal of the SeMiWa [50], a situation reasoner within an ubiquitous, assisted live environment should forecast situations based on the current classified one. The ranking algorithm is an adaption of the one presented in this article. The *factual visualization knowledge* is conceptually similar to *factual lifecycle knowledge*, such as circadian or infradian rhythms. The *domain assignment* changes the prediction based on the current *domain* where is user is situated, e. g., at home, at work, or outside. The *user and device* information are identical since they consist of personalized context information for one user, combined with the sensoral context of the surrounding environment. An important additional benefit is the usage of *user-shared, empirical knowledge*, like it is mentioned in this work. Therefore, we also propose a collaborative filtering approach for finding "neighbors" that are acting in a similar way.

Acknowledgment

This work is partly funded by the German Federal Ministry of Education and Research under promotional reference number 01IA09001C; and by the European Social Fund and the Federal State of Saxony in Germany within project VICCI (ESF-100098171). Furthermore, the authors wish to thank Michael Aleythe for implementation support.

References

[1] M. Voigt, S. Pietschmann, L. Grammel, and K. Meißner, "Context-aware recommendation of visualization components," in *Proc. of the 4th International Conference on Information, Process, and Knowledge Management (eKNOW 2012)*. XPS, Feb. 2012, pp. 101–109.

[2] R. Haber and D. A. McNabb, "Visualization idioms: A conceptual model for scientific visualization systems," *Visualization in Scientific Computing*, pp. 74–93, 1990.

[3] L. Grammel and M.-A. Storey, "Choosel - web-based visualization construction and coordination for information visualization novices," in *Proc. of. IEEE InfoVis 2010*, 2010.

[4] M. Chen, D. Ebert, H. Hagen, R. Laramee, R. van Liere, K.-L. Ma, W. Ribarsky, G. Scheuermann, and D. Silver, "Data, information, and knowledge in visualization," *Computer Graphics and Applications, IEEE*, vol. 29, no. 1, pp. 12 –19, Jan. 2009.

[5] J. Heer, F. van Ham, S. Carpendale, C. Weaver, and P. Isenberg, *Creation and Collaboration: Engaging New Audiences for Information Visualization*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 92–133.

[6] M. Voigt, S. Pietschmann, and K. Meißner, "A semantics-based, end-user-centered information visualization process for semantic web data," in *Semantic Models for Adaptive Interactive Systems*, T. Hussein, H. Paulheim, S. Lukosch, J. Ziegler, and G. Calvary, Eds. Springer Berlin / Heidelberg, 2013, pp. 81–106.

[7] M. Chen and H. Hagen, "Guest editors' introduction: Knowledge-assisted visualization," *Computer Graphics and Applications, IEEE*, vol. 30, no. 1, pp. 15 –16, jan.-feb. 2010.

[8] J. Mackinlay, "Automating the design of graphical presentations of relational information," *ACM Trans. Graph.*, vol. 5, no. 2, pp. 110–141, 1986.

[9] S. MacNeil and N. Elmqvist, "Visualization mosaics for multivariate visual exploration," *Computer Graphics Forum*, vol. 0, pp. 1–15, 2013.

[10] O. Gilson, N. Silva, P. Grant, and M. Chen, "From web data to visualization via ontology mapping," in *Computer Graphics Forum*, vol. 27, no. 3. Blackwell Publishing Ltd, Sep 2008, pp. 959–966.

[11] S. M. Casner, "Task-analytic approach to the automated design of graphic presentations," *ACM Trans. Graph.*, vol. 10, pp. 111–151, April 1991.

[12] S. F. Roth and J. Mattis, "Automating the presentation of information," in *Artificial Intelligence Applications, 1991. Proc. of 7th IEEE Conf. on*, vol. 1. IEEE, 1991, pp. 90–97.

[13] J. Mackinlay, P. Hanrahan, and C. Stolte, "Show me: Automatic presentation for visual analysis," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1137–1144, Nov. 2007.

[14] D. Gotz and Z. Wen, "Behavior-driven visualization recommendation," in *IUI '09: Proceedings of the 13th international conference on Intelligent user interfaces*. New York, NY, USA: ACM, 2009, pp. 315–324.

[15] X. Wang, D. H. Jeong, W. Dou, S.-W. Lee, W. Ribarsky, and R. Chang, "Defining and applying knowledge conversion processes to a visual analytics system," *Computers & Graphics*, vol. 33, no. 5, pp. 616 – 623, 2009.

[16] J. J. van Wijk, "The value of visualization," in *Proceedings of IEEE Visualization*, Minneapolis, MN, October 2005, pp. 79–86.

[17] G. Shu, N. Avis, and O. Rana, "Bringing semantics to visualization services," *Advances in Engineering Software*, vol. 39, no. 6, pp. 514–520, 2008.

[18] M. G. Skjveland, "Sgvizler: A javascript wrapper for easy visualization of sparql result sets," in *ESWC 2012*, 2012.

[19] S. Few, *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly Media, Inc., 2006.

[20] C. Cappiello, M. Matera, M. Picozzi, G. Sprega, D. Barbagallo, and C. Francalanci, "Dashmash: A mashup environment for end user development," in *Web Engineering*, ser. Lecture Notes in Computer Science, S. Auer, O. Daz, and G. Papadopoulos, Eds. Springer Berlin Heidelberg, 2011, vol. 6757, pp. 152–166.

[21] D. Duke, K. Brodlie, D. Duce, and I. Herman, "Do you see what i mean? [data visualization]," *Computer Graphics and Applications, IEEE*, vol. 25, no. 3, pp. 6–9, May 2005.

[22] R. Potter and H. Wright, "An ontological approach to visualization resource management," in *DSV-IS*, 2006, pp. 151–156.

[23] P. Rhodes, E. Kraemer, and B. Reed, "Vision: an interactive visualization ontology," in *ACM-SE 44: Proceedings of the 44th annual Southeast regional conference*. New York, NY, USA: ACM, 2006, pp. 405–410.

[24] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D. McGuinness, E. Sirin, and N. Srinivasan, "Bringing semantics to web services with owl-s," *World Wide Web*, vol. 10, pp. 243–277, Sep. 2007.

[25] D. Fensel, M. Kerrigan, and M. Zaremba, *Implementing Semantic Web Services: The SESA Framework*. Springer, 2008.

[26] J. Farrell and H. Lausen, "Semantic annotations for WSDL and XML Schema," http://www.w3.org/TR/sawsdl/, W3C, Aug. 2007.

[27] J. Kopecký and T. Vitvar, "Wsmo-lite: Lowering the semantic web services barrier with modular and light-weight annotations," in *Proc. of the Intl. Conf. on Semantic Computing*, Aug. 2008, pp. 238–244.

[28] Y. Chabeb, S. Tata, and A. Ozanne, "YASA-M: A semantic web service matchmaker," *Proc. of the Intl. Conf. on Advanced Information Networking and Applications*, pp. 966–973, Apr. 2010.

[29] F. Gilles, V. Hoyer, T. Janner, and K. Stanoevska-Slabeva, "Lightweight composition of ad-hoc enterprise-class applications with context-aware enterprise mashups," in *Proc. of the Intl. Conf. on Service-Oriented Computing*. Springer, 2009, pp. 509–519.

[30] P. Lops, M. Gemmis, and G. Semeraro, "Content-based recommender systems: State of the art and trends," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer US, 2011, pp. 73–105.

[31] R. Bambini, P. Cremonesi, and R. Turrin, "A recommender system for an iptv service provider: a real large-scale production environment," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer US, 2011, pp. 299–331.

[32] D. M. Nichols, "Implicit rating and filtering," in *In Proceedings of the Fifth DELOS Workshop on Filtering and Collaborative Filtering*, 1998, pp. 31–36.

[33] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, ser. CSCW '94. New York, NY, USA: ACM, 1994, pp. 175–186.

[34] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, ser. WWW '01. New York, NY, USA: ACM, 2001, pp. 285–295.

[35] C. Desrosiers and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. Springer US, 2011, pp. 107–144.

[36] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," 2007.

[37] M. Voigt and J. Polowinski, "Towards a unifying visualization ontology," TU Dresden, Institut fuer Software und Multimediatechnik, Dresden, Germany, Technical Report TUD-FI11-01, Mar. 2011, ISSN: 1430-211X.

[38] J. Polowinski and M. Voigt, "VISO: A shared, formal knowledge base as a foundation for semi-automatic infovis systems," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems 2013 (CHI'13)*, 2013.

[39] V. Tietz, G. Blichmann, S. Pietschmann, and K. Meiner, "Task-based recommendation of mashup components," in *Proceedings of the 3rd International Workshop on Lightweight Integration on the Web (ComposableWeb 2011)*. Springer, Jun. 2011.

[40] D. Gotz and M. Zhou, "Characterizing users' visual analytic activity for insight provenance," in *Visual Analytics Science and Technology, 2008. VAST '08. IEEE Symposium on*, Oct. 2008, pp. 123 –130.

[41] A. Mitschick, S. Pietschmann, and K. Meißner, "An ontology-based, cross-application context modeling and management service," *Intl. Journal on Semantic Web and Information Systems (IJSWIS)*, Feb. 2010.

[42] W. S. Cleveland and R. McGill, "Graphical perception: Theory, experimentation, and application to the development of graphical methods," *Journal of the American Statistical Association*, vol. 79, no. 387, pp. 531–554, 1984.

[43] D. Lin, "An information-theoretic definition of similarity," in *Proc. 15th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1998, pp. 296–304.

[44] M. R. McLaughlin and J. L. Herlocker, "A collaborative filtering algorithm and evaluation metric that accurately model the user experience," in *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '04. New York, NY, USA: ACM, 2004, pp. 329–336.

[45] S. Pietschmann, C. Radeck, and K. Meißner, "Semantics-based discovery, selection and mediation for presentation-oriented mashups," in *Proceedings of the 5th International Workshop on Web APIs and Service Mashups*, ser. ACM ICPS. ACM, Sep. 2011.

[46] S. Pietschmann, A. Mitschick, R. Winkler, and K. Meißner, "CroCo: Ontology-Based, Cross-Application Context Management," in *Proceedings of the 3rd International Workshop on Semantic Media Adaptation and Personalization (SMAP 2008)*. Prague, Czech Republic: IEEE CPS, Dec. 2008.

[47] M. Voigt, V. Tietz, N. Piccolotto, and K. Meißner, "Attract me! how could end-users identify interesting resources?" in *Proceedings of the 3rd International Conference on Web Intelligence, Mining and Semantics (WIMS'13)*, no. 978-1-4503-18. ACM, Jun. 2013.

[48] M. Voigt, A. Werstler, J. Polowinski, and K. Meißner, "Weighted faceted browsing for characteristics-based visualization selection through end users," in *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, ser. EICS '12. New York, NY, USA: ACM, 2012, pp. 151–156.

[49] M. Voigt, A. Mitschick, and J. Schulz, "Yet another triple store benchmark? Practical experiences with real-world data," in *Proc. of. the 2nd International Workshop on Semantic Digital Archives (SDA)*, 2012.

[50] M. Franke, C. Seidl, and T. Schlegel, "A seamless integration, semantic middleware for cyber-physical systems," in *10th International Conference on Networking, Sensing and Control 2013*, 2013.