Demonstration of a Customizable Knowledge Graph Visualization Framework*

Vitalis Wiens[1,2] and Steffen Lohmann[3]

[1] L3S Research Center, Leibniz University of Hannover, Germany
wiens@l3s.de
[2] TIB Leibniz Information Centre for Science and Technology, Hannover, Germany
[3] Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS St. Augustin, Germany

Abstract. In the context of the Semantic Web, various visualization methods and tools exist. However, suitable visualizations are highly dependent on individual use cases and targeted user groups. Therefore, existing solutions require modifications and adjustments to meet the demands of other use cases and user groups. In this demo, we present an approach for a unified framework addressing customizable visual representations of knowledge graphs. Our approach refines the commonly used steps in the visualization generation process (i.e., data access, mapping to visual primitives, and rendering) for Semantic Web contexts. Separation of concerns for individual steps and a modular and customizable architecture build the foundation for a pipeline-based visualization framework. The framework enables the creation and selection of *the right components for the right tasks*, realizing a variety of use cases and visual representations in Semantic Web contexts.

Keywords: Ontology visualization, knowledge graph visualization, customization, visual representation, visualization pipeline framework.

1 Introduction

In Semantic Web contexts, various visualization methods and tools are available, and new ones are being developed every year. The applied visualization methods range from indented trees and chord diagrams to treemaps and Euler diagrams. According to a recent survey [1], most applications use two-dimensional graph-based representations in the form of node-link diagrams. Furthermore, this survey indicates that "*new visualization methods and tools are often developed from scratch, omitting opportunities to learn from previous mistakes or to reuse advanced techniques provided by other researchers and developers.*"

The number of visualization methods, tailor-suited tools, along with the requirements and necessity for customization indicate that a *one-size-fits-all* solution is challenging, if not impossible, nor feasible, to realize. This particular

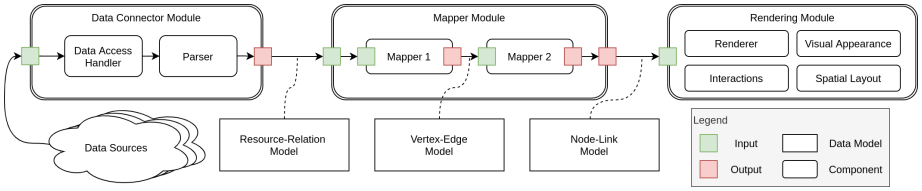

Fig. 1. An example pipeline using components for creating node-link diagrams.

observation, and the fact that applications are often developed from scratch, motivates the design for a unified visualization framework approach. The challenge of unification is realised through divergence of different components, allowing for adjustments and customization for the visualization of the data at hand. This conceptualization allows to create various visualizations through the exchange of components for individual use cases and data source. As our approach targets a unified visualization framework approach, we address this through the divergence of customizable components and convergence in data models.

Our approach builds on the identification of commonly used steps in the creation of visual representations for the Semantic Web. Furthermore, the approach employs the paradigm of separation of concerns for individual steps, increasing its flexibility. Our framework realizes the approach and provides a modular and customizable visualization pipeline that serves as a foundation for creating visual representations of knowledge graphs for different data sources, use cases, and user groups. Customizable components serve as stand-alone artifacts for different steps in the creation process of visualizations. The framework organizes the different components in a pipeline. The data models, provided as input and output for various components, serve as convergence points within the visualization pipeline. Therefore, we argue that the data models further contribute to the reusability of components and sections of visualization pipelines. Figure 1 illustrates an example pipeline realizing node-link diagrams.

2 Approach

Most visualization tools employ the following steps to create visual representations: **i)** access the data; **ii)** map it to visual primitives (e.g., geometric shapes); **iii)** render the primitives; and **iv)** optionally add animations and user interactions. Typically, these steps are created within an application addressing a single use case and a single user group. In order to address the various requirements of different use cases and user groups, our approach employs a separation of concern paradigm and refines the visualization generation process.

We argue that due to the variety of Semantic Web technologies, visualization methods, and tools, a unified visualization framework is achievable through divergence. Components perform specialized tasks, whereas their results are converged into corresponding data models. The refined visualization process focusing on the creation of node-link diagrams is presented in Table 1.

Step	Component	Responsibility
1	Data Access Handler	Specify Semantic Web data source and parameters for data retrieval in JSON format.
2	Parser	Process retrieved Semantic Web data and organize it in the `Resource-Relation Model`.
3	Vertex-Edge Mapper	Select data for visual representation and create graph structure.
4	Node-Link Mapper	Modify the graph structure using merge, split, and aggregation operations.
5	Rendering	Create visual primitives and integrate the specifications of other components (6–8).
6	Visual Appearance	Specify how nodes and links are rendered.
7	Spatial Layout	Specify how elements are organized in the layout.
8	Interactions	Specify interactions for graph, nodes, and links.

Table 1. Tabular representations of different components and their responsibilities.

3 Customizable Visualization Pipeline Framework

The visualization generation process typically involves the identification of data sources, its mappings to visual primitives, and optionally the creation of user interactions. Our framework[4] provides a customizable visualization pipeline, allowing users to configure data sources, apply graph manipulations, and define the visual appearance of rendering elements. This conceptualization allows to create various visualizations through the exchange of components for individual use cases and data sources.

In this demo, the pipeline can access four example data sources. The node-link mappers demonstrate different graph manipulations. The rendering module creates node-link diagram visualizations that are adjustable w.r.t. their visual appearance that is based on GizMO [2]. Additionally, the rendering module implements basic user interactions such as zooming, dragging and panning. A force-directed layout is used to create the spatial assignment of nodes and links in the graph visualization.

The framework provides basic implementations for individual components. All data models and the components for the data access module and the mapper module are implemented in plain JavaScript. The rendering components use D3.js additionally for creating interactions and visual primitives as SVG elements. Furthermore, this framework allows users to export the pipeline as a bundle which includes the implementation of the components and a React application frame. We argue that the bundle serves as a basic infrastructure for extending and adjusting the implementation to the requirements of different use cases and user groups. Figure 2 gives an overview of the UI of the framework.

[4] `https://github.com/vitalis-wiens/donatello-pipelines`

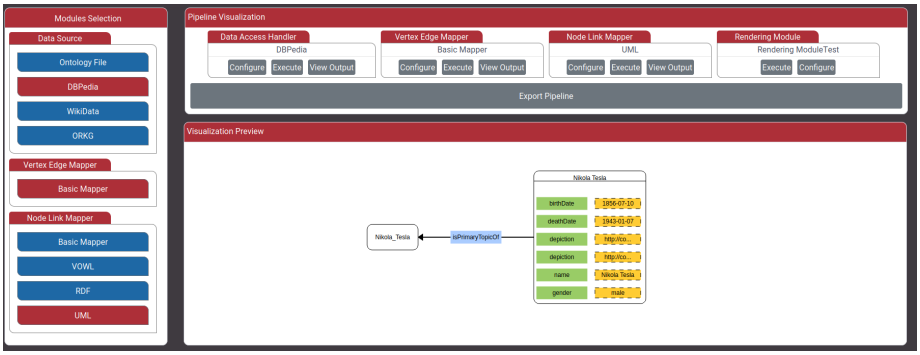

Fig. 2. Overview of the UI of the framework. Left side: Module selection for data sources, vertex-edge mapper, and node-link mapper. *Note: In this version we implemented only a single vertex-edge mapper.* Top: Pipeline configuration. Bottom: Visualization preview based on the selected components and their configurations.

4 Discussion

The limitations of the current prototype implementation for this demonstration are twofold. First, the UI for creating pipelines is fixed w.r.t. to the arrangement of the components, i.e., data-source selection, vertex-edge mapper, node-link mapper, and rendering module. Second, our prototype implementation addresses only the visual representations in the form of node-link diagrams. Thus, the data models and the pipeline composition are tailor-suited for such visualizations.

Furthermore, the components use a top-down communication where their results feed into the next component in the pipeline. We envision the bottom-up communications to enable the creation of applications that allow through means of visual editing also to modify the data source. For example, changes in the graph propagate backward through the pipeline and create updates for the data source, e.g., SPARQL query updates, REST-API calls for creating new data, or integration with version control systems such as git.

Our approach exports the created pipeline as a zip file creating source code for a fully functional React core application that serves as an entrance point for development. Separation of concerns for individual components and the pipeline organization allow developers to adjust and customize individual components to their needs and the requirements of the underlying use case. Due to the aspects that our components are derived through the separation of concern paradigm, and their conceptualization as stand-alone artifacts, (i.e., each component takes inputs, process them, and provides one output), allow for their reuse in other visualizations. The implementation of our components uses class inheritance, thus new components can be derived from existing ones that serve as examples.

5 Demonstration

In the demonstration, we will introduce the approach realizing customizable visualization pipelines in Semantic Web contexts. While our demo application targets to showcase the flexibility and ease-of-use to create visualizations in the form of node-link diagrams, its current implementation is a prototype, realizing the necessary configurations and modifications in pre-selected components. We will introduce the individual components, their responsibilities, their interplay, and how to extend and configure individual parts of the pipeline to create visual representations for knowledge graphs. Furthermore, we will explain how the pipeline configuration is used to create a bundle comprising of the implemented components, the pipeline configuration, its initialization, and a core React application framework.

At ISWC, we will provide a hands-on experience of the demo application: Users will be enabled to create custom visualization pipelines using the configuration UI of the application. The configured pipelines can be downloaded as zip files, thus users are equipped with an infrastructure to directly start developing and adjusting the pipeline to their needs. We will direct the audience to the demo web application, allowing for independent testing. The application is open source and available under the MIT license, thus allowing for reusing, extending, and contributing to the project.

Finally, we hope that discussions with Semantic Web experts in the context of the conference will allow us to identify further requirements, needs, and features for the approach and its implementation. An overview of the application, its features, and usage is illustrated in the corresponding demo video[5].

Acknowledgments

This work is co-funded by the European Research Council project ScienceGRAPH (Grant agreement #819536).

References

1. Dudáš, M., Lohmann, S., Svátek, V., Pavlov, D.: Ontology visualization methods and tools: a survey of the state of the art. Knowledge Eng. Review **33** (2018)
2. Wiens, V., Lohmann, S., Auer, S.: Gizmo–a customizable representation model for graph-based visualizations of ontologies. In: Proceedings of the 10th International Conference on Knowledge Capture. pp. 163–170 (2019)

[5] `https://drive.google.com/file/d/17cSLjDNq7kpepfbmZiYevM10eWUE9gRh/preview`